RICE UNIVERSITY

Discrete Search Optimization for Real-Time Path Planning in Satellites

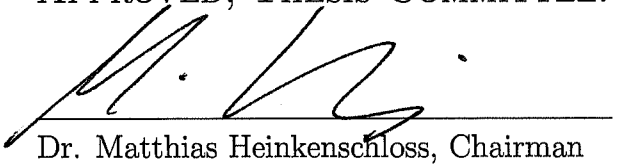
by

Millie Mays

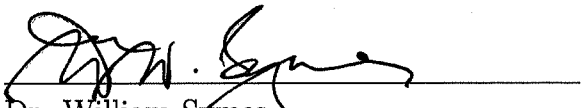
A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Arts

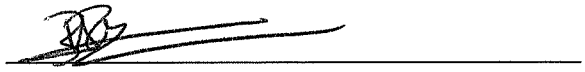
Approved, Thesis Committee:



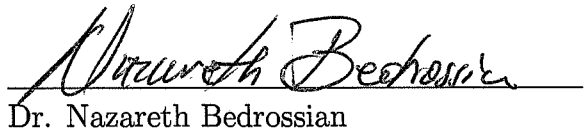
Dr. Matthias Heinkenschloss, Chairman
Professor of Computational and Applied Mathematics



Dr. William Symes
Noah Harding Professor of Computational and Applied Mathematics



Dr. Beatrice Riviere
Associate Professor of Computational and Applied Mathematics



Dr. Nazareth Bedrossian
Group Lead, Manned Space Systems,
Charles Stark Draper Laboratory, Inc.

Houston, Texas

April, 2012

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Abstract

Discrete Search Optimization for Real-Time Path Planning in Satellites

by

Millie Mays

This study develops a discrete search-based optimization method for path planning in a highly nonlinear dynamical system. The method enables real-time trajectory improvement and singular configuration avoidance in satellite rotation using Control Moment Gyroscopes. By streamlining a legacy optimization method and combining it with a local singularity management scheme, this optimization method reduces the computational burden and advances the capability of satellites to make autonomous look-ahead decisions in real-time. Current optimization methods plan offline before uploading to the satellite and experience high sensitivity to disturbances. Local methods confer autonomy to the satellite but use only blind decision-making to avoid singularities. This thesis' method seeks near-optimal trajectories which balance between the optimal trajectories found using computationally intensive offline solvers

and the minimal computational burden of non-optimal local solvers. The new method enables autonomous guidance capability for satellites using discretization and stage division to minimize the computational burden of real-time optimization.

Acknowledgements

I have to admit that getting the Draper Fellowship to study at Rice University was nothing short of a miracle, and so first of all I must thank God for arranging it.

I am very grateful to Dr. Nazareth Bedrossian and the Charles Stark Draper Laboratory for providing the Draper Lab Fellowship, giving me such a challenging problem, and having faith that I could accomplish something with it. Thanks to "Naz" for teaching me that tough skin is the key to air power and to Sagar Bhatt for helping me bridge the gap between math and engineering. I am equally grateful to Dr. Matthias Heinkenschloss and Rice University for the opportunity to earn a master's degree and for your guidance and mentorship throughout the process. Thank you for keeping me grounded and reminding me that I am a mathematician, not an engineer.

My husband, Ralph Hale, has kept me calm and sane throughout this process; thank you for your patience. Thanks to my parents Mauri and Mark, for believing in me and teaching me to dream big. Mariah, you are the best sister I could wish for.

Thanks to all my friends here in Houston and to the members of the Rice Graduate Christian Fellowship for your prayers and support throughout this journey; I wish all of you the best. Jayme Yeo, thank you for being my running buddy and giving me "talk therapy" Monday, Wednesday, and Friday mornings for two years straight.

Finally, I'm grateful to the United States Air Force for allowing me to attend graduate school for the first two years of my career.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis proposes a search-based optimization method to manage Control Moment Gyroscopes (CMGs) to avoid singular states. My method enables real-time improvement of rotation trajectories for satellites.

In 1991, Joseph Paradiso demonstrated the feasibility of a search-based method for controlling arrays of CMGs to manage satellite momentum in order to execute precise rotation trajectories [22]. Since Paradiso's method required user interaction to define the desired momentum profile, his search-based method was classified as an "offline" solution method, since it could not be employed autonomously by a satellite. His approach sought near-optimal trajectories in a limited global sense, since the momentum profile was limited to the one defined by the user; therefore, it seemed inferior to advanced solutions to the global optimization problem obtained using two-point boundary value solvers.

Still, neither the two-point boundary value solvers nor Paradiso's search method was simple enough to be employed autonomously by an actual satellite. For years, the focus has shifted to improving the "local" solvers - those that guide the satellite using a local linearization of the complex and highly nonlinear control problem solved by Paradiso and the other "global" solvers. Such methods include singularity-robust variations of the pseudoinverse ([1], [8], [16]), local gradient methods [1], and limiting constraints on the capabilities of the gyroscopes [9]. However, all the local solvers lose the benefit of "look-ahead," or predicting what the local choices will do to the satellite's trajectory in the future. Their advantage is that they are easy to employ autonomously on a satellite.

This thesis revisits Paradiso's goal of enabling a satellite to make local decisions using global information - in other words, giving a satellite the autonomous capability to make "look-ahead" decisions like a global solver. To accomplish this, I distill Paradiso's search method into a more simplified and computationally efficient form, and set it in a real-time framework which enables the satellite to optimize a small portion of its upcoming trajectory using global information about the effect of its choice upon the entire trajectory.

I succeed in demonstrating the feasibility of the search-based method for real-time application using a battery of Monte Carlo simulations. In nearly all cases, the search-based method succeeds in accomplishing the satellite's mission while improving its trajectory in the receding stage division method of [12].

In Chapter 2, I provide a high-level overview of the existing methods of utilizing Control Moment Gyroscopes to control a satellite. By investigating the benefits and limitations of some of the most prominent methods developed by other researchers, I frame the motivation for the trajectory selection algorithm proposed by this thesis.

With the background of existing research in place, in Chapter 3 I present the mathematical formulation of the CMG problem. This chapter includes necessary variable definitions, explanations of spacecraft attitude dynamics, presentation of the differential equations involved in the satellite motion, and the use of inverse kinematics to select the appropriate gyroscope motion. Finally, I present the general optimal control problem and the simplified form of the optimal control problem which, in its discrete form, is the basis for the method proposed in this thesis.

After the mathematical problem formulation is presented in Chapter 3, in Chapter 4 I revisit the existing research in order to more effectively compare the framework of established methods with the new search-based method proposed by this thesis. I review two different choices for singularity measure, and several variants of the pseudoinverse in order to explain the differences between several local solution methods and a few global solution methods. Of particular importance is the review of Paradiso's search-based method from [22] because it is the parent of the method presented in this thesis.

In Chapter 5 I discretize the optimal control problem of Chapter 3 in order to use a graphical approach to improving the satellite trajectory. I explain why natural

candidates for solving discrete problems cannot be applied to the CMG problem; neither Dynamic Programming nor Shortest Path Planning can be applied in this case. Finally, I present the simplified search-based optimization method which I distilled from Paradiso's original search-based method. The simplification modifications enable the fast computation necessary to apply the method as a real-time trajectory improvement scheme.

In Chapter 6, I develop two unique frameworks to apply my search-based method for real-time application. I also present the results of hundreds of random simulations using various conditions to test the robustness of the search-based method for finding near-optimal trajectories. In some of the simulations, I restrict the computation time allowed for the search to simulate real-time trajectory selection to establish the feasibility of the stage division scheme with my search-based method. In nearly all cases, the search-based method finds a trajectory which successfully accomplishes the satellite's mission, demonstrating that the method is feasible for real-time implementation.

Chapter 2

Problem Background

This thesis proposes a discrete search-based method by combining previously existing avoidance and escape methods into an improved hybrid which provides predictive guidance to the satellite's internal gyroscopes to manage singular configurations in satellite rotation. By combining recently developed singular direction robust inverse kinematics with a well-established search-based method, the algorithm developed in this thesis advances the potential for intelligent predictive trajectory guidance decisions in real time.

This chapter describes the type of gyroscopes which will be studied in this thesis, explains why singular configurations are an issue for successful gyroscope rotation guidance, and provides an overview of the three existing types of steering logics: singularity avoidance, singularity escape, and hybrid avoidance and escape. This thesis proposes a new steering logic which fills the need for a hybrid method which a

satellite can use to make predictive guidance decisions to manage the Control Moment Gyroscopes effectively.

2.1 Control Moment Gyroscopes

A control moment gyroscope (CMG) is a momentum storage device used to control the orientation and rotation of spacecraft. An array of several CMGs is preferred to manage the momentum of a satellite because unlike external gas-propulsion thrusters which eventually run out of fuel, CMGs are internal to the spacecraft and run on electricity continuously provided by solar power. An array of 3 CMGs was used onboard the U.S. space station Skylab (1973-1979) [5, p. 67], and an array of 4 98 kg double-gimbal CMGs is in use onboard the International Space Station [3, p. 7]. The CMGs have been used most frequently in large spacecraft, where the cost advantage of using electrically-powered attitude control is larger due to the prohibitive cost of using fuel-propellant thrusters alone to move something as large as a space station [16]. However, CMGs are being effectively "miniaturized" in both size and cost to make them feasible for smaller satellite missions. An example is the 28 kg CMG produced by Honeywell [13]. CMG arrays allow the satellite to accomplish precision pointing and tracking missions such as earth imaging and precision global positioning, laser communications, and scientific experiments including star mapping and astronomical research [16].

There are several types of CMGs, only one of which will be examined in detail in

this report; the single-gimbal CMG (SGCMG) is the least expensive, least mechanically complex, and the least heavy of any of the CMGs. A special property of any CMG is called torque amplification, whereby commanding a small torque on a CMG, a large torque is provided to the satellite; this means that CMGs are highly efficient at managing satellite momentum. Although all CMGs have this property to some degree, the SGCMG provides the best torque amplification and is therefore the most efficient [16].

Although an array of SGCMGs is the most desirable for satellite application because of its high torque amplification, high reliability and low cost, the SGCMG has one significant drawback compared to the other types of CMGs, and it is related to the number of degrees of freedom each type of CMG provides to the control system. The SGCMG has only one gimbal axis, and therefore only one controllable degree of freedom by rotating the CMG about this axis. See figure 2.1 for an image of an SGCMG. The heavy wheel spins at a constant rate, providing momentum along its axis of spin. The direction of the momentum vector is controlled by changing the gimbal angle θ of the SGCMG about the gimbal axis. The SGCMG is only able to produce momentum orthogonal to the gimbal axis.

For comparison, I note that Double-Gimbal CMGs and Variable-Speed CMGs have two degrees of freedom. Double-Gimbal CMGs have two independent gimbal axes whereas Variable-Speed CMGS have only one gimbal axis but can accelerate the flywheel and change its speed [16]. The number of degrees of freedom is important

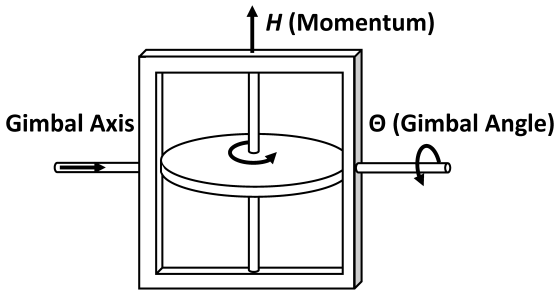


Figure 2.1: The flywheel spins at a constant rate about the momentum axis, while the control law directs θ to change the direction of the momentum vector. The CMG is incapable of producing momentum in the direction of the gimbal axis.[1]

to the satellite system design because it provides redundancy in case of mechanical failure; it also allows for the possibility of careful steering to avoid singularities. However, reducing the number and size of the CMGs on board a satellite reduces cost and weight. Skylab was built with only 3 CMGs, and one failed during Skylab's mission, "making the task of adjusting the spacecraft's attitude considerably more complicated" [5, p. 66]. Modern applications in general have arrays of more CMGs to provide greater degrees of freedom and redundancy. With 4 double-gimbal CMGs, the International Space Station has 8 degrees of freedom [26].

Satellite arrays such as the one used in this thesis, with 4 SGCMGs and therefore only 4 degrees of freedom, are referred to as "minimally redundant" [1]. These arrays are advantageous because of reduced cost, size, weight, and mechanical complexity compared to larger arrays of CMGs with more degrees of freedom. However, with only 1 degree of freedom available for singularity avoidance, the task of steering the satellite

to avoid or minimize the impact of singular configurations is much more challenging! This thesis proposes discrete search-based guidance to avoid singularities which can cause mission failure in a precision pointing or tracking mission.

2.2 Singularities

The major drawback of the SGCMG, and the reason that it was not selected for use on the International Space Station despite its advantages, is the existence of configurations of the CMG gimbal angles where it is impossible for the array of CMGs to provide the desired torque to the satellite. Such a configuration is referred to as a "singularity." In a targeting or precision pointing mission, reaching such a singularity could mean mission failure. To avoid singularities or minimize the damage they cause, an intelligent guidance algorithm such as the one developed in this thesis must direct the movement of the gimbal angles to avoid singular configurations of the satellite system.

There are two main types of singularities: external and internal. External singularities, also called saturation singularities, occur when all of the CMG momentum vectors in a system are projected maximally in a certain direction. Then, no more torque can be produced in that direction because the satellite is already producing all the possible torque in that direction. The volume bounded by these external singularities is known as the "momentum envelope" and is shown in figure 2.2. For a satellite with 4 SGCMGs arranged in a pyramid, the momentum envelope can be

thought of as a near-spherical ball with two dimples on the gimbal axis of each CMG. The reason it is not spherical is because, to produce torque aligned with the gimbal axis of CMG 1, that CMG cannot contribute since it can only produce angular momentum orthogonal to the desired torque. In order to prevent the satellite from spinning, CMG 3 on the opposite pyramid face must point its angular momentum vector opposite that of CMG 1.

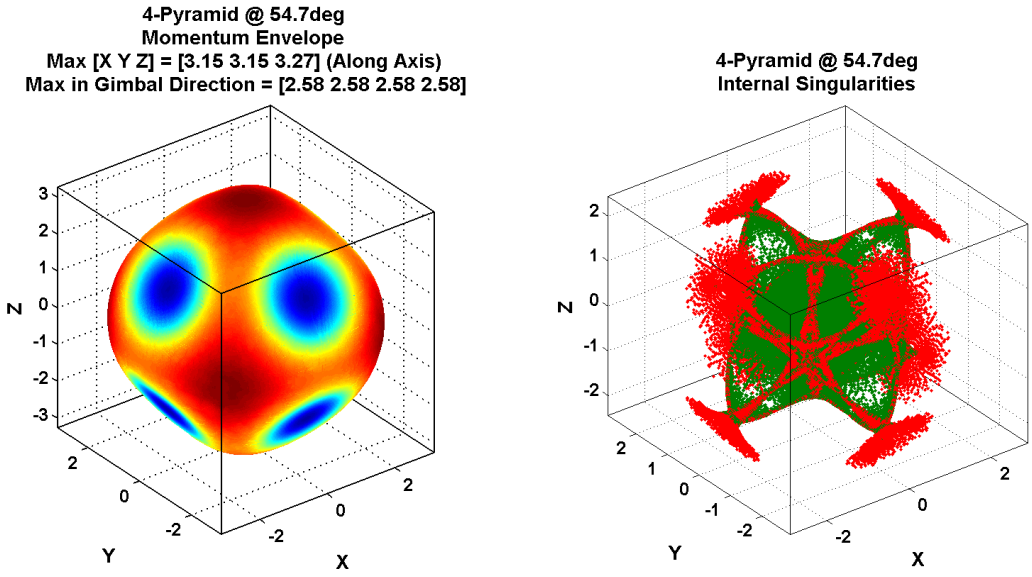


Figure 2.2: Left: For 4-SGCMGs pyramid mount with $\beta = 54.7^o$, the momentum envelope shows the maximum momentum that can be achieved using the CMGs. Dimples occur at the axis of rotation of each CMG. Right: Internal Singularities inside the momentum envelope which correspond to a singular configuration of the CMGs. Adapted from [17],[22].

The non-spherical nature of the momentum envelope is one issue which makes

singularity management more complicated. One simple way that has been employed to manage singularities is to use larger SGCMGs than are necessary for the satellite mission, so that the external singularities on the dimpled momentum envelope are far from the maximum momentum needed to complete the satellite's mission [22]. However, this adds unnecessary cost and weight to the satellite.

The second type of singularity is located inside the momentum envelope and is called an internal singularity; see figure 2.2. This type of singularity occurs at configurations of the gimbal angles such that the momentum vectors of all the SGCMGs lie within a plane. In such a configuration of the CMGs, there is instantaneously a direction in which it is impossible to produce torque [17]. Mathematically, they are discernible when the Jacobian of the momentum does not have full rank. The singular direction is left vector associated with the zero singular value in the singular value decomposition of the Jacobian. Therefore, a function of the singular values can be used to generate a "measure" of singularity; the traditional choice in the literature is to use a normalized determinant of the Jacobian multiplied by its transpose to give an indication of its rank: $\det(JJ^T)$. This can be thought of like a distance measure. When the determinant is greater than 1, the satellite configuration is "far" from singularity; as the determinant approaches 0, the satellite configuration is "near" singularity, and when the determinant equals 0 the satellite is in a singular state.

Margulies and Aubrun, in their original publication laying out the theory of CMGs, developed both a geometric and an analytic way to classify the types of singularities

[17]. Their derivation provided the necessary background framework for the development of many of the CMG steering laws reviewed in this chapter, and of this thesis. In particular, Margulies and Aubrun provided the framework for the use of null motion. When a CMG system has more than 3 degrees of freedom, as in the satellite system used in this thesis which has 4 SGCMGs and therefore 4 degrees of freedom, then it is possible to move the gimbals in such a way that no net torque is produced [17].

The use of null motion is critical to many types of steering laws which attempt to avoid singular configurations. Margulies and Aubrun provided the original proof that of the two types of internal singularities, hyperbolic and elliptic, only "hyperbolic" singularities can be avoided using null motion. To escape an "elliptic" singularity, it is necessary to add undesirable torque error to the system [17]. It is always preferable to avoid singularities rather than to escape them using torque error; however, Margulies and Aubrun showed that sometimes it is impossible to avoid a singularity and so adding torque error is inevitable. Margulies and Aubrun's classification of singularity types was the basis of the singularity type-specific Hybrid Steering Law of Leve and Fitz-Coy in 2010 [16], where avoidance and escape are applied equally often. In the algorithm proposed in this thesis, avoidance through null motion is always prioritized, and escape via torque error occurs only as a last resort. This is only possible because the proposed method is a unique hybrid of the methods which have already been developed in the satellite industry.

2.3 Existing Steering Logics

A steering logic is an algorithm or rule which enables the satellite to intelligently choose the gimbal rates to use to achieve the requested torque command. There are three main types of steering logics which have been applied to the SGCMG problem: singularity avoidance, singularity escape, and hybrid algorithms which can both avoid singularities when possible and escape singularities when necessary [16].

There are six terms which are used to classify different types of steering laws. A *local* method takes information about where the satellite is *right now* and uses it to make a decision about where to go next. It is named because the method only has access to *local* information. In contrast, a *global* method has complete information about where the satellite will start, where it is going, and where it might go in between. A global method can employ all of this data to decide what the satellite should do throughout a trajectory.

A *real-time* method makes decisions which are immediately or very quickly implemented by the satellite. A real-time method gathers information as the satellite is flying in order to make decisions about what it should do *right now*. Its opposite is an *offline* method which uses information which is assumed to be known far in advance to tell the satellite what to do well before the satellite starts doing it.

Finally, an *autonomous* method is built-in to the satellite. The satellite can make all its guidance decisions without any input from computers or humans on the ground. A *non-autonomous* method interacts with a computer or human system outside of

the satellite.

Some of these terms are used synonymously. In general, local methods require little computational power and so they are generally both autonomous and real-time. Similarly, global methods usually require a great deal of computational resources and so a global method is usually performed by computers on the ground and uploaded to the satellite; so the method is both non-autonomous and offline. In a recent review of existing steering logics, it was noted in [16] that singularity avoidance algorithms are either local in nature and calculated real-time or they are global in nature and the trajectories are calculated offline.

This thesis challenges such conventional wisdom by proposing a hybrid avoidance and escape method which is also a hybrid between the local and global methods. The algorithm provides a middle ground by using global information to make local decisions in order to find near-optimal trajectories without the high computational requirements of the fully global methods. This local-global hybrid can be termed "predictive" or "look-ahead" and it enables the satellite to make autonomous, real-time decisions by combining an existing global hybrid avoidance and escape algorithm with a local singularity escape algorithm.

2.3.1 Singularity Avoidance Algorithms

Singularity avoidance algorithms work by using null motion or by constraining the set of usable gimbal angles to be singularity-free. An example of constrained gimbal

angles is the recently patented "scissored pairs" method of singularity avoidance. In this method, pairs of SGCMGs operate much like scissors, with the angle between them changing on command. To prevent singularities, it is necessary only to put a non-zero minimum on the angle between the SGCMGs; then, it is impossible for the momentum vectors to all lie within a plane, and no singularity is encountered [9]. By avoiding singularities or utilizing only singularity-free areas of the momentum envelope, these algorithms do not have to deviate from the desired mission path in order to escape internal singularities. This is particularly important in high-precision pointing applications. However, they do reduce the usable work space of the momentum envelope, which means that larger-than-necessary CMG arrays must be used to carry out mission requirements. The algorithm developed by this thesis will not apply these types of artificial constraints, so the entire available momentum envelope is used, and cost is reduced by closely matching the mission requirements to the size of the CMG array.

Global offline solvers such as 2-point boundary value solvers can find trajectories which avoid singularities and such methods are well-developed for CMG application but the highly nonlinear nature of the problem means that the offline solution is extremely sensitive to disturbances in initial conditions. It is also highly desirable in the field of CMG research to develop autonomous steering laws which do not require the input of human interaction or computational resources from the ground. Current satellite technology is not capable of using a 2-point boundary value solver in real-

time.

Local singularity avoidance laws are autonomous, but do not have the advantage of predictive decision-making, since they solve a linearized local version of the problem. As Bedrossian noted in [1], "...A general defect of locally-optimal procedures which was ... they tend to lock into trajectories of locally maximum gain which can actually lead to singular configurations."

One such local method which attempts to operate as a singularity avoidance method is the Local Gradient (LG) method. Recall that the Jacobian of the momentum can be used to create a function to measure the "distance" of the system from singularity. Then the LG method uses null motion to stay on a local optimum of this function, applying the projection of the gradient of the determinant into the null space of the Jacobian in order to maximize the "distance" of the system from singularity. Because of the Jacobian is 3×4, the null space always has dimension of at least 1. The LG method is analogous to a hiker trying to stay at as high an altitude as possible (far from singularity); in order to do so, he will walk along the top of a ridge line. However, Bedrossian showed that the LG method is easily drawn down into an elliptic singularity while staying on a local optimum [1]. In the hiking analogy, this means that the hiker is staying on the crest of the ridge but the altitude of the ridge is slowly decreasing. Eventually, the hiker ends up in a valley (a singular state) even though he was staying on top of the ridge the whole time. This makes the LG method less than ideal as an option for singularity avoidance.

It is important to note that to achieve a near-optimal trajectory and stay away from singularities, it is sometimes necessary to make a decision which, on a local level, seems sub-optimal in the sense that it causes the system to temporarily approach a singularity. However, by so doing, the overall distance from singularity over the course of the trajectory can be improved. In terms of our analogy, the hiker should leave the ridge earlier in order to get on a ridge or a mountain whose altitude will not eventually lead him to singularity. This basic issue with local singularity avoidance methods is the reason this thesis proposes a look-ahead hybrid method, which will investigate multiple choices of null motion so that a non-optimal local decision will be allowed if it ultimately leads to a better trajectory.

2.3.2 Singularity Escape Algorithms

Singularity escape algorithms are probably the simplest of all steering laws. Local in nature, they do not even attempt to avoid singularities through null motion. Instead, they use some variant of the pseudoinverse solution to calculate the gimbal rates necessary to achieve a desired torque command. When a singularity is approached, these algorithms deviate from the desired mission path in order to maintain the rank of the Jacobian and thereby to "escape" the singularity without suffering its ill-effects. Although the introduced torque error is undesirable for precision and pointing missions, the singularity robust nature of these algorithms means that the satellite is able to control itself real-time and autonomously with little to no offline input.

Singularity robust methods of computing the inverse kinematics were first proposed for the field of robotics manipulation by Nakamura and Hanafusa in 1986 [19]. The direct analogy between robotics manipulation and CMG momentum management schemes was developed by Bedrossian in [1], who proposed the singularity-robust (SR) inverse which is commonly used in CMG applications as an alternative to the general Moore-Penrose pseudoinverse.

In 2000, Ford and Hall demonstrated that while the SR inverse adds undesirable torque error in the directions of all three eigenvectors of the Jacobian, it is only necessary to add torque error in the direction of the left vector associated with the smallest singular value in the singular value decomposition of the Jacobian in order to maintain the pseudoinvertibility. To that end, they developed the Singular Direction Avoidance (SDA) inverse as an improved alternative to the SR inverse for precision pointing. They successfully showed in [8] that the SDA inverse adds significantly less undesirable torque error than the SR inverse, while still keeping the Jacobian at full rank.

This thesis proposes a hybrid escape-and-avoidance method, and the SDA method is the variant of the pseudoinverse which will be employed as the escape mechanism in the algorithm. However, both the SR inverse of Bedrossian and the SDA inverse of Ford and Hall are incapable of any intelligent predictive avoidance of singularity because all of the information is local. As Bedrossian noted in [1], a singularity robust variant of the pseudoinverse is incapable of avoiding a singularity without an

appropriate null motion algorithm. In this thesis, the SDA singularity escape method will be combined with a search-based null-motion singularity avoidance method to create a singularity-robust hybrid.

2.3.3 Hybrid Avoidance and Escape

The type of algorithm developed in this thesis is a singularity avoidance and escape method, which employs a hybrid approach by mixing the singularity robustness of the escape algorithms with the null motion of the avoidance algorithms to maximize precision pointing and tracking. It searches for nonsingular trajectories but will transverse a singular region if mission requirements call for speed rather than precision pointing and the singularity cannot be easily avoided.

Leve and Fitz-Coy recently developed a local hybrid method which they call Hybrid Steering Logic (HSL) [16]. Using the singularity classification method developed by Margulies and Aubrun [17], who developed much of the theoretical and analytical framework for CMG research, Leve and Fitz-Coy determine whether a singularity is "hyperbolic" or "elliptic." If the system approaches a hyperbolic singularity, it can be avoided via null motion without introducing any unwanted torque error. If the system approaches an elliptic singularity, torque error is applied to escape the singularity and null motion, which Leve and Fitz-Coy consider less useful, is suppressed. Therefore Leve and Fitz-Coy developed unique scalar metrics to scale null motion and torque error appropriately depending on the type of singularity that is being approached

[16].

A disadvantage of the HSL method is that like the singularity escape algorithms, HSL is a local method without any capability for predictive decision-making. Additionally, the choice for the null motion algorithm which they combined with the state-of-the-art escape method, SDA inverse, is questionable. They chose to employ the Local Gradient method which was shown by Bedrossian in [1] to frequently lead to singularity rather than avoiding it (see subsection 2.3.1). Their decision to suppress null motion when in the neighborhood of an elliptic singularity is also unsupported. Although it was shown by Margulies and Aubrun [17] that null motion *at* an elliptic singularity cannot move the system out of singularity, Bedrossian showed in [1] that the most significant effect of the SR-inverse (and presumably, the SDA inverse as well) applied *near* elliptic singularity is to slow the system response to *enable* the application of null motion. However, Leve and Fitz-Coy reduce null motion when in the vicinity of an elliptic singularity, even when the null motion could still be used to avoid the singular state.

Finally, Leve and Fitz-Coy do not identify one of the scalars necessary to implement their method, except to define what scale the scalar is on. It is unclear how this scalar should be selected and so it is difficult to duplicate or verify their work. However, in spite of all of these drawbacks it is still apparent from the success of the HSL method that a more promising combination of null motion, applied at both hyperbolic and elliptic singularities, and the SDA method for singularity escape, could be a very

successful hybrid. The algorithm developed in this thesis follows this heritage.

A second avoidance-and-escape algorithm which is the foundation of the method proposed in this thesis is the directed search developed by Paradiso in [22]. Although labeled by Leve and Fitz-Coy [16] as a singularity avoidance global offline solver, Paradiso's intention in developing the method was to create a framework which could eventually enable real-time autonomous predictive decision making possible for the satellite. This is sometimes referred to as "look-ahead" capability, and his goal is the same as that of this thesis. The reason Paradiso's work has fallen into the global category instead of forging a path towards "look-ahead" capability is because his algorithm had a user interface to define the desired momentum profile, even though he intended for the search-based method to become an autonomous steering capability for a satellite [22].

In this thesis, to further the goal of creating a intelligent predictive autonomous decision-making strategy for the satellite, the desired momentum profile is defined autonomously by the algorithm. The algorithm in its most basic form is a distilled form of Paradiso's. His use of different cost functions to evaluate the partial trajectories and the complete trajectories has been replaced by a single, simplified cost function for computational efficiency. However, the framework of the discrete search space and the policy for choosing new starting nodes for a new trajectory is identical to Paradiso's original proposal.

This thesis utilizes Paradiso's discrete search framework. Paradiso discretizes the

system dynamics and makes a discrete graph of potential trajectories which would accomplish the satellite's mission, with each node symbolizing a particular decision between different values of null motion which must be made by the satellite. By allowing various options for null motion instead of requiring the locally optimal solution as did Leve and Fitz-Coy, Paradiso uses predictive decision making to avoid the trap of being pulled into a singularity while remaining at a locally optimal solution [22]. His discrete formulation of a graph-theoretic approach to trajectory optimization and some terms of his cost function used to choose between the potential nodes in a trajectory are both used in this thesis. Several of the terms of the cost function which Paradiso needed in order to make the method feasible on a slow Macintosh II have been removed. The second cost function, employed by Paradiso to compare trajectories, was also removed for computational efficiency in favor of using only one cost function. Finally, this thesis updates Paradiso's method by applying the improved SDA inverse which was first proposed 10 years after Paradiso originally showed that the discrete search was feasible using the SR inverse.

The main contribution of this thesis is the real-time framework for applying the search method. While Paradiso's algorithm could 'improve' the trajectory in under a minute or conduct a complete search in 5-10 minutes, real-time application requires that the algorithm improve the trajectory in a matter of seconds [22, p. 62]. To apply it real-time, I will make use of research from a different field of trajectory optimization. Ikaida, et al. developed a method which is somewhat reminiscent of the

receding horizon strategy of control theory. They called this method "stage division" and applied it to optimizing helicopter landing trajectories for noise reduction. By dividing the time horizon into stages, Ikaida et al. proposed that a global offline solver could be applied online as a "look-ahead" solver [12]. At each stage, the algorithm is expected only to improve upon the currently selected trajectory. The optimization occurs in real time, so at the end of each stage, the trajectory is updated and the helicopter - or the satellite - follows the currently selected trajectory while the algorithm continues to search for improvements. After calculating a trajectory from t_0 to t_{f1} and having the satellite begin to follow it at t_0, the algorithm works to calculate the next stage of the trajectory from t_{f1} to t_{f2} and finishes the selection of the near-optimal trajectory before the satellite finishes the first trajectory at t_{f1}. Using this method, as long as the algorithm calculation is capable of being computed rapidly enough by the satellite itself, a "global offline" solver can be applied online as a "look-ahead" solver.

The onboard computational capabilities of current satellite technologies do not allow other global offline solvers such as the 2-point boundary value solvers to be computed by the satellite in real-time. However, Paradiso's method of discretizing the decision-space of the satellite reduces the computational complexity of the problem even though the optimization is still being carried out in a more global sense. This allowed this thesis to solve the discretized system as a real-time look ahead method.

Paradiso's method is presented in a framework of feasibility; the major claim of his

report is to demonstrate that the search-based method is feasible. Therefore, while he does explain the inspiration for his search method (waypoint planning), he does not really explore the theoretical support for his approach. However, his formulation of the problem is similar to a subset of the shortest path planning problem where the costs of edges are not known in advance. This is known as the Canadian Traveler Problem, where knowledge of the edge costs in a graph is achieved by exploration, which also has an associated cost [21]. One of the contributions of this thesis is to formulate the search-based approach of Paradiso in terms of a graph-theoretical shortest-path problem in order to extend the search-based approach to a wider field of study.

General solutions for the Canadian Traveler Problem are not known, but Karger and Nikolova developed an optimal strategy to find a near-optimal solution while minimizing the cost of exploration for perfect binary trees with edge costs of 0 or 1 assigned in a Bernoulli distribution [14]. Their optimal strategy calls for exploring all the paths whose exploration cost is 0, and then returning to the node which is closest to the goal, and which therefore has the highest potential to be in the optimal trajectory, and continuing the search from that node using the same strategy after traversing the edge with cost 1 [14].

The crossover to the discrete search-based method of this thesis is not exact, but the discrete search space can be formulated as a so-called "perfect tertiary tree" similar to the perfect binary tree of Karger and Nikolova. While my discrete search

tree has no edges of cost 0, my strategy to search for a near-optimal trajectory is remarkably similar to the optimal strategy devised by Karger and Nikolova: explore a few trajectories with low calculation (exploration) cost to gather information about the tree; then choose the node with the lowest cost value, and which therefore has the highest potential to be on the optimal trajectory, and continuing the search from that node using the same strategy until a near-optimal trajectory is identified. So although it was published 17 years after Paradiso demonstrated the "feasibility" of the search-based approach, the work of Karger and Nikolova provides a good theoretical basis for my policy of exploring the unknown costs of the discrete search space. On the basis of this support, the algorithm developed in this thesis updates the search-based approach with advances in CMG research made by the local solvers.

This thesis develops and tests an algorithm which identifies near-optimal trajectories by using a discrete search which combines several existing methods. Karger and Nikolova's "optimal policy" for the solution of the Canadian traveler problem on perfect binary trees [14] is combined with with the "search-based approach" of Paradiso [22], which is improved by using the SDA inverse of Ford and Hall [8] for singularity escape. By applying this algorithm real-time using the stage division method of Ikaida et al., this algorithm enables real-time predictive decision making on the part of the satellite, for minimally redundant systems of SGCMGs.

Chapter 3

Problem Description

The major purpose of research in the field of Control Moment Gyroscope (CMG) guidance is to find a reliable algorithm to steer a satellite from an initial state $\mathbf{e}_0$ to a final state $\mathbf{e}_f$. In section 3.1 I describe the variables and differential equations necessary to describe the motion of the satellite. In section 3.2 I present the objective function which will be used in the general optimal control problem (OCP) defined in section 3.3 when the particular rotation to transition from $\mathbf{e}_0$ to $\mathbf{e}_f$ is unconstrained.

In section 3.4 I make the first assumption of the solution method of this thesis by supposing that the satellite has a mission requirement to accurately track a particular rotation profile which is defined in advance. Although the particular rotation will depend on the mission requirements, for the purposes of this thesis the algorithm will attempt to track a smooth continuous, sinusoidal attitude profile $\mathbf{e}(t)$ defined in appendix A. Under this assumption, I describe the inverse kinematics which my

solution method will utilize to minimize the computational complexity required for the satellite to compute solution trajectories, and present a simplified OCP created by adding the path constraint.

Finally, in section 3.5 I create the penalty form of the OCP used in this thesis. This simple OCP will be the basis of the search-based method proposed by this thesis for autonomous use by satellites to avoid singular configurations in their rotation trajectories in real time.

3.1 Satellite Dynamics

The satellite motion dynamics will be described using a set of three ordinary differential equations. These equations require several variables: first, the body rate $\omega : \mathbb{R} \to \mathbb{R}^3$ describes how the satellite is rotating about 3 axes, roll, pitch, and yaw. The second is the vector of gimbal angles $\theta : \mathbb{R} \to \mathbb{R}^4$ which gives the angles of the gimbals of each CMG and describes the *configuration* of the CMG array. The variable $\theta(t)$ is expressed in terms of angular change from a neutral or starting CMG configuration. The difference between ω and θ is shown visually in figure 3.1. The third state variable describes the attitude of the satellite.

The attitude of a satellite can be expressed in several ways. In the basic form of the optimal control program, it is convenient to express attitude in terms of Euler Parameters (often referred to in the literature as quaternions) $q : \mathbb{R} \to \mathbb{R}^4$. Each quaternion can be described by an angular rotation $\phi(t) \in \mathbb{R}$ about an axis of rotation

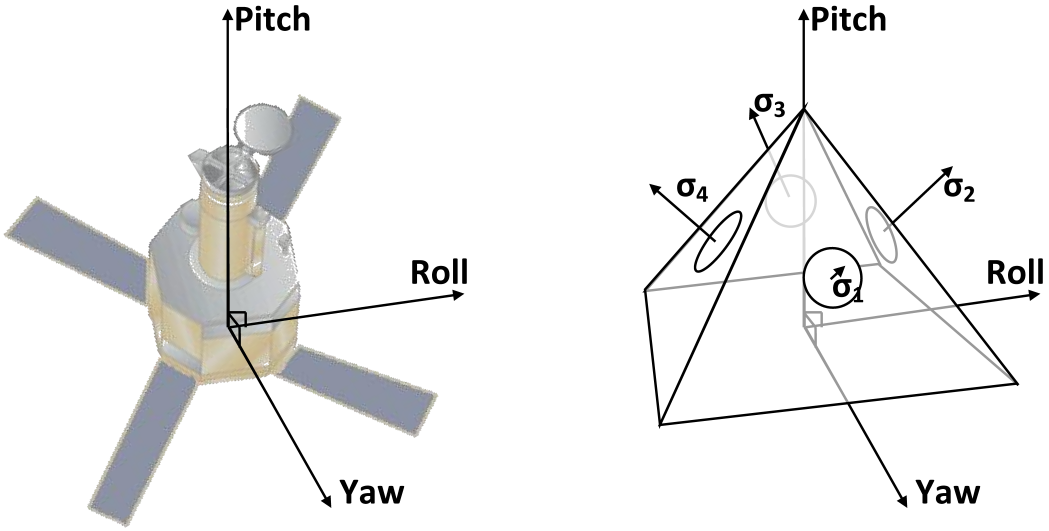


Figure 3.1: Body Frame of the Satellite. The variable ω describes motion about the Roll, Pitch, and Yaw axes while θ describes motion about the gimbal axes σ_i. TRACE satellite image from [20].

defined by the unit vector $\hat{e}(t) \in \mathbb{R}^3$ as follows:

$$q(t) = \begin{bmatrix} q_0(t) \\ q_1(t) \\ q_2(t) \\ q_3(t) \end{bmatrix} = \begin{bmatrix} cos(\frac{\phi(t)}{2}) \\ \widehat{e}_i(t)\sin(\frac{\phi(t)}{2}) \\ \widehat{e}_j(t)\sin(\frac{\phi(t)}{2}) \\ \widehat{e}_k(t)\sin(\frac{\phi(t)}{2}) \end{bmatrix}. \tag{3.1}$$

The parameter $\mathbf{I} \in \mathbb{R}^{3\times3}$ represents the inertia of the satellite. The matrix used

in this thesis is similar to that of a TRACE-like satellite [28, p. 440] and is given by

$$\mathbf{I} = \begin{bmatrix} 36.09 & 0.42 & 0.24 \\ 0.42 & 54.72 & 0.24 \\ 0.24 & 0.24 & 57.27 \end{bmatrix} \text{ kg-m}^2. \tag{3.2}$$

Then using these state variables and parameters, the satellite rotation dynamics are defined using the rigid body dynamics derived in [10],[11]:

$$\dot{q}(t) = \frac{1}{2} T(q(t)) \left[0 \; \omega(t)^T\right]^T, \tag{3.3}$$

$$\dot{\omega}(t) = -\mathbf{I}^{-1} \left[\omega(t) \times (\mathbf{I}\omega(t) + H(\theta(t))) - \dot{H}(\theta(t))\right], \tag{3.4}$$

where

$$\dot{H}(\theta(t)) \equiv \frac{d}{dt} H(\theta(t)) = J(\theta(t))\dot{\theta}(t). \tag{3.5}$$

In these equations $\dot{\omega}(t)$ is the rotational acceleration of the satellite. The variable $\dot{q}(t)$ is the rate of change of attitude of the satellite, $\dot{H}(\theta(t))$ is the torque generated by the CMGs, and $\dot{\theta}(t)$ is the rate of change of the gimbal angles of the CMGs in the satellite. Later $\dot{\theta}$ will be used to control the satellite. The matrix $T(q(t))$ is the quaternion transformation matrix

$$T(q(t)) = \begin{bmatrix} q_0(t) & -q_1(t) & -q_2(t) & -q_3(t) \\ q_1(t) & q_0(t) & -q_3(t) & q_2(t) \\ q_2(t) & q_3(t) & q_0(t) & -q_1(t) \\ q_3(t) & -q_2(t) & q_1(t) & q_0(t) \end{bmatrix}. \tag{3.6}$$

The total momentum of the CMGs $H(\theta) : \mathbb{R}^4 \to \mathbb{R}^3$ depends on $\theta(t)$ and the configuration of the satellite as defined in equation (3.7). The total momentum is

the vector sum of the momentum of each CMG. In this case the satellite under investigation has 4 single gimbal CMGs in a "4-pyramid" structure with a skew angle of $\beta = 54.7^o$ as shown in figure 3.2. Both $\omega(t)$ and $H(\theta(t))$ are expressed in the body reference frame of the satellite. For the satellite under consideration,

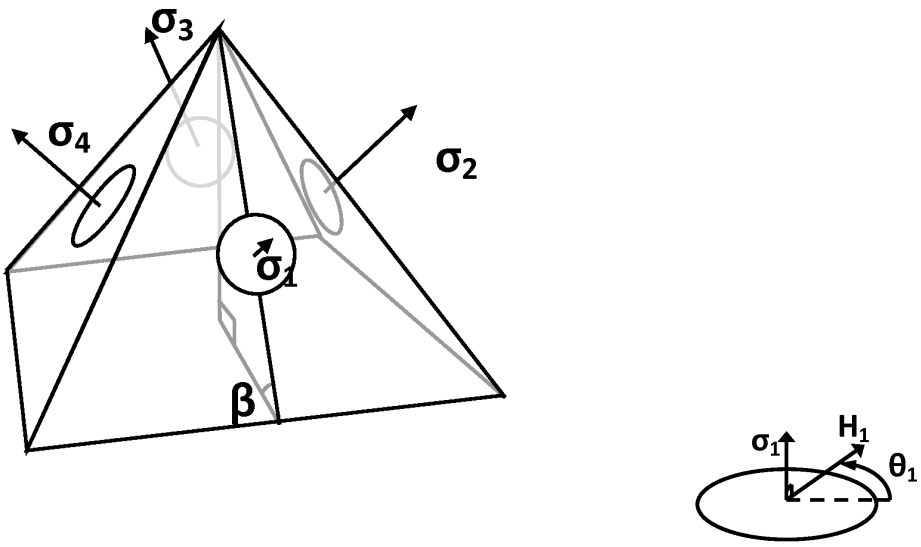


Figure 3.2: The 4-SGCMGs pyramid mount with $\beta = 54.7^o$. The momentum vector H_i sweeps out a momentum plane perpendicular to the gimbal axis $\sigma_i$and its direction is determined by θ_i.

$$\frac{H(\theta(t))}{h_{CMG}} = \begin{bmatrix} -c_\beta \sin(\theta_1(t)) \\ \cos(\theta_1(t)) \\ s_\beta \sin(\theta_1(t)) \end{bmatrix} + \begin{bmatrix} -\cos(\theta_2(t)) \\ -c_\beta \sin(\theta_2(t)) \\ s_\beta \sin(\theta_2(t)) \end{bmatrix} + \begin{bmatrix} c_\beta \sin(\theta_3(t)) \\ -\cos(\theta_3(t)) \\ s_\beta \sin(\theta_3(t)) \end{bmatrix} + \begin{bmatrix} \cos(\theta_4(t)) \\ c_\beta \sin(\theta_4(t)) \\ s_\beta \sin(\theta_4(t)) \end{bmatrix} \tag{3.7}$$

where $c_\beta = \cos(\beta)$ and $s_\beta = \sin(\beta)$, and $h_{CMG} \in \mathbb{R}$ is the magnitude of each CMG momentum vector. Throughout this thesis, the value of h_{CMG} is assumed to be the same for each CMG. Consequently, the Jacobian of the momentum is

$$\frac{J(\theta(t))}{h_{CMG}} = \begin{bmatrix} -c_\beta \cos(\theta_1(t)) & \sin(\theta_2(t)) & c_\beta \cos(\theta_3(t)) & -\sin(\theta_4(t)) \\ -\sin(\theta_1(t)) & -c_\beta \cos(\theta_2(t)) & \sin(\theta_3(t)) & c_\beta \cos(\theta_4(t)) \\ s_\beta \cos(\theta_1(t)) & s_\beta \cos(\theta_2(t)) & s_\beta \cos(\theta_3(t)) & s_\beta \cos(\theta_4(t)) \end{bmatrix}. \tag{3.8}$$

The Jacobian is used to define the cost function, using an equation for what is known as "CMG gain" in the literature.

3.2 CMG Gain and Cost Function

Since the Jacobian is needed to compute the torque of the satellite system (3.5), it can also be used to create a measure of the current capacity of the satellite system to produce torque in any direction. This is known in the literature as "CMG gain", so-called because when the system has no "gain" there is a direction in which the CMGs cannot produce torque [17]. Such a configuration of CMGs is called a singularity, and

the main purpose of the algorithm developed in this thesis is to guide the satellite to avoid such states. First I will define the equation for CMG gain and then I will define the cost function which will be employed in this thesis.

The satellite is in a singular configuration if the Jacobian $J(\theta(t)) \in \mathbb{R}^{3\times 4}$ has rank less than three. In that case it is impossible to produce torque in the direction of the left singular vector of the Jacobian associated with the zero singular value. CMG gain is traditionally defined using the determinant as a way to measure the "distance" from singularity of the Jacobian, and is given by

$$m(\theta(t)) = \sqrt{|J(\theta(t))J(\theta(t))^T/h_{CMG}^2|}, \tag{3.9}$$

where $|\cdot|$ is the determinant and h_{CMG} is the individual momentum of each CMG flywheel. For the purposes of this thesis, I assume that all four CMGs have the same momentum. Scaling by h_{CMG} is necessary because, as Ford and Hall noted in [8], the value of $\sqrt{|J(\theta(t))J(\theta(t))^T|}$ varies with the size of the system because its units are $(\text{N-m-s})^2$, or squared angular momentum. The determinant is subject to scaling. If $\epsilon \in \mathbb{R}$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ then

$$|\epsilon A| = \epsilon^n |A|.$$

When the determinant is normalized to a unitless measure of singularity as in (3.9), the non dimensional value of $m(\theta(t))$, would be the same for any 4-pyramid satellite with a skew angle of $\beta = 54.7^o$, and so it is consistent to compare the relative success of various steering laws using this singularity measure.

CMG gain values range from 0 to just above 1. This imprecise range was a major

argument of Ford and Hall in [8] for a new measure of singularity which they developed which had lower and upper bounds of 0 and 1, respectively, and which will be discussed in section 4.1. However, their measure has not been adopted in the literature and so any comparison between methods is simplified by using the traditional determinant measure. From my survey of the literature and existing methods I generalize that values above 1 are "very good" while values above 0.2 are "acceptable" and those below 0.1 are considered to be "near-singular." Values much higher than 1 are not necessarily desirable because "what goes up must come down" and high values of CMG gain are not always sustainable.

Using the equation for CMG gain, I now present the cost function which will be used to define the general optimal control problem for this problem.

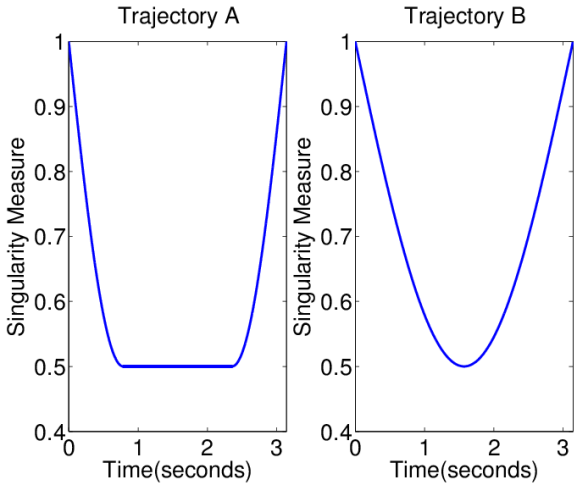


Figure 3.3: Because of the integral objective function defined in the first term of equation (3.10), trajectory B is superior to trajectory A.

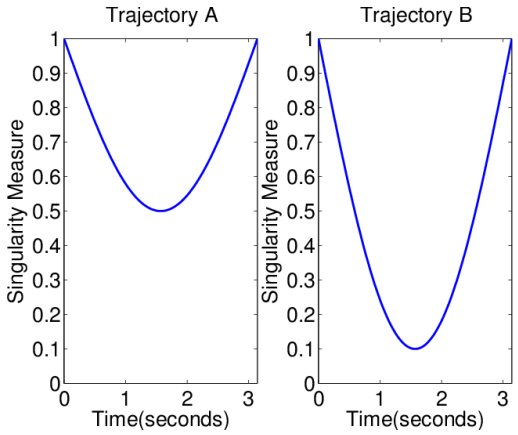


Figure 3.4: Because of the final time objective function defined in the second term of equation (3.10), trajectory A is superior to trajectory B.

The cost function is based on the cost function employed by Paradiso [22] and has two parts: an integral objective designed to optimize the average value of CMG gain over the trajectory (figure 3.3), and a final time objective designed to maximize the lowest value of CMG gain over the trajectory (figure 3.4). The cost function is

$$\min \left[W_1 \int_{t_0}^{t_f} \frac{1}{m(\theta(t))} dt - W_2 \min_{t \in [0, t_f]} m(\theta(t)) \right], \tag{3.10}$$

where W_1 and W_2 are weights used to scale the relative importance of each type of objective. This function tries to make the minimum CMG gain over the trajectory as high as possible, and also works to minimize the amount of time spent in low-gain states.

The first term of equation (3.10) is designed to penalize any trajectory which lingers for a period of time close to a singular state. This means that a trajectory which dips quickly through a low-gain state is preferable to one which stays in a

low-gain state for a long period of time. A simple comparison of two trajectories is shown in figure 3.3. Although the worst-case gain of the two trajectories is the same, trajectory B passes through the low-gain state more quickly than trajectory A, which maintains a state of low gain for a longer period of time.

The second term of equation (3.10) works to maximize the lowest value of CMG gain over the trajectory. A simple comparison example is shown in figure 3.4. Trajectory A is favored by the optimal control program because the lowest value of CMG gain over trajectory A is higher than the lowest value of the CMG gain of trajectory B.

In the next section I present the general form of the OCP used to describe the CMG problem, using the objective function defined in this section.

3.3 General Optimal Control Problem

Using the system dynamics of (3.3) and (3.4) and the objective function from (3.10), the general form of the optimal control problem (OCP) under consideration transitions from initial states q_0, ω_0, and θ_0 to final states q_f, and ω_f via states $q(t)$, $\omega(t)$,

and $\theta(t)$ while satisfying hardware constraints on $\dot{\theta}(t)$ for $t \in [t_0,\ t_f]$, and is given by

$$\min_{\omega,q,\theta,\vec{u}} \left[W_1 \int_{t_0}^{t_f} \frac{1}{m(\theta(t))} dt - W_2 \min_{t\in[0,t_f]} m(\theta(t)) \right] \tag{3.11a}$$

s.t.

$$\begin{bmatrix} \dot{\omega}(t) \\ \dot{q}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} -\mathbf{I}^{-1} \left[\omega(t) \times (\mathbf{I}\omega(t) + H(\theta(t))) - \dot{H}(\theta(t)) \right] \\ \frac{1}{2} T(q(t)) \left[0\ \omega(t)^T \right]^T \\ \vec{u}(t) \end{bmatrix} \tag{3.11b}$$

$$\begin{bmatrix} \omega(t_0) - \omega_0 \\ q(t_0) - q_0 \\ \theta(t_0) - \theta_0 \end{bmatrix} = 0 \tag{3.11c}$$

$$\begin{bmatrix} \omega(t_f) - \omega_f \\ q(t_f) - q_f \end{bmatrix} = 0 \tag{3.11d}$$

$$\left\| \dot{\theta}(t) \right\|_\infty - \dot{\theta}_{max} \leq 0, \quad t \in [t_0, t_f]. \tag{3.11e}$$

There are many solution methods to find good trajectories using this OCP (see for example [2]). Unfortunately most satellites do not have the capability to solve this OCP using the limited onboard computational capacity.

In the next section I will assume that a desired torque profile $\dot{H}(t)_{desired}$ has been preselected in order to use the inverse kinematics to reduce computational complexity by simplifying the optimal control problem.

3.4 Mission Assumption and Inverse Kinematics

This thesis makes a basic assumption about the requirements of the satellite's mission in order to develop a simplified OCP which will be the basis for the discrete optimization I propose. Although the OCP defined in (3.11) has a path constraint on the control variable $u(t)$, there are no path constraints on the state variables. However, satellites with precise tracking missions have specific path constraints telling them what attitude they must have at each moment [27, p. 3].

In this thesis, I assume that the satellite is required to follow a specific attitude profile from an initial state $\mathbf{e}_0$ to a final state $\mathbf{e}_f$ defined in Euler Angles (a product of rotation matrices about the yaw, pitch, and roll axes). Although the particular rotation will depend on the mission requirements, for the purposes of this thesis the algorithm will attempt to track a smooth continuous, sinusoidal attitude profile defined in Appendix A. In the Appendix I explain the spacecraft attitude dynamics necessary to use $\mathbf{e}(t)_{desired}$ and $\dot{\mathbf{e}}(t)_{desired}$ to generate the required body rate and attitude profiles, $\omega(t)_{desired}$ and $\dot{\omega}(t)_{desired}$.

Using these desired profiles and the differential equation given in (3.4), I calculate the desired momentum $H(t)_{desired}$ and torque $\dot{H}(t)_{desired}$ profiles. The value of $H(t_0)$ is known at the start of the calculation. Therefore in order to accomplish the satellite mission, all that is necessary is to closely track the desired torque profile, generated by

$$\dot{H}(t)_{desired} = -\mathbf{I}\dot{\omega(t)}_{desired} - (\omega(t)_{desired} \times (\mathbf{I}\omega(t)_{desired} + H(t)_{desired})). \tag{3.12}$$

The purpose of the algorithm developed in this thesis is to calculate values of $\dot{\theta}(t)$ and $\theta(t)$ which closely track the desired torque profile $\dot{H}(t)_{desired}$. The optimization will try to minimize the difference between the desired and actual torque profiles as shown below. Using the definition of the Jacobian of the momentum given in (3.5), we solve for the conditions under which

$$\dot{H}(t)_{desired} = \dot{H}(\theta(t)) = J(\theta(t))\dot{\theta}(t). \tag{3.13}$$

Note that the Jacobian of the system is 3×4 and therefore not invertible, and so there is no unique solution for $\dot{\theta}(t)$. If the Jacobian has full rank it is possible to compute one particular solution for $\dot{\theta}(t)$ using the Moore-Penrose pseudo inverse given by

$$J(\theta(t))^{\dagger} = J(\theta(t))^{T}(J(\theta(t))J(\theta(t))^{T})^{-1}. \tag{3.14}$$

See [17].

When the smallest singular value of the Jacobian approaches 0, the satellite system is near a singular configuration. Then commanded gimbal rates computed using the pseudoinverse solution approach infinity. When the rank of the Jacobian actually equals 2, then

$$R\Big(J(\theta(t))\Big) \subsetneq \mathbb{R}^3$$

and for

$$\dot{H}(t)_{desired} \notin R\Big(J(\theta(t))\Big)$$

there exists no $\dot{\theta}(t)$ that satisfies (3.13).

This is the reason methods to solve the CMG problem work extensively to maintain the rank of the Jacobian. Usually in the literature, equation (3.14) or some other variant of the pseudoinverse is used for the inverse kinematics to calculate $\dot{\theta}(t)$. In the literature the pseudoinverse solution $\dot{\theta}(t)^{\dagger}$ is often referred to as the particular solution and is given by

$$\dot{\theta}(t)^{\dagger} = J(\theta(t))^{\dagger}\dot{H}(t)_{desired}. \tag{3.15}$$

This thesis employs a modification of the pseudoinverse, defined in equation (4.5), to compute the particular solution.

The general solution comes by using the null space of the Jacobian. The possibility of null motion was introduced by Margulies and Aubrun in [17]. When the Jacobian has rank 3, define $\dot{\theta}_{null}(t) \in N\Big(J(\theta(t))\Big)$ where $N\Big(J(\theta(t)\Big)$ is the null space of the Jacobian. Then since by definition $u(t)J(\theta(t))\dot{\theta}_{null}(t) = 0$ for any scalar $u(t) \in \mathbb{R}$ we can use the vector $u(t)\dot{\theta}_{null}(t)$ to create the general solution

$$\dot{\theta}(t) = J(\theta(t))^{\dagger}\dot{H}(t)_{desired} + u(t)\dot{\theta}_{null}(t). \tag{3.16}$$

If the satellite is disturbed or there is error in tracking $\dot{H}(t)_{desired}$, it is desirable to use an updated quantity in place of $\dot{H}(t)_{desired}$ to correct for the disturbance. To do so I make use of the original system dynamics (3.4) and the desired body rate and acceleration profiles $\omega(t)_{desired}$ and $\dot{\omega}(t)_{desired}$, but replace $H(t)_{desired}$ with the actual momentum of the system. Then the updated general solution is given by

$$\dot{\theta}(t) = J(\theta(t))^{\dagger}\dot{H}(\theta(t))_{update} + u(t)\dot{\theta}_{null}(t), \tag{3.17}$$

where

$$\dot{H}(\theta(t))_{update} = -\mathbf{I}\dot{\omega}(t)_{desired} - (\omega(t)_{desired} \times (\mathbf{I}\omega(t)_{desired} + H(\theta(t)))). \tag{3.18}$$

In equations (3.16) and (3.17), the control variable is explicitly stated as $u(t)$. Under the assumption that the satellite is required to follow a specific attitude profile, the OCP (3.11) is therefore replaced by

$$\min_{\theta,u} \left[W_1 \int_{t_0}^{t_f} \frac{1}{m(\theta(t))} dt - W_2 \min_{t\in[0,t_f]} m(\theta(t)) \right] \tag{3.19a}$$

$$\text{s.t.} \quad \dot{\theta}(t) = J(\theta(t))^{\dagger} \dot{H}(\theta(t))_{update} + u(t)\dot{\theta}_{null}(t), \quad t \in [t_0, t_f], \tag{3.19b}$$

$$\theta(t_0) - \theta_0 = 0 \tag{3.19c}$$

$$\dot{H}(t)_{desired} = J(\theta(t))\dot{\theta}(t), \quad t \in [t_0, t_f], \tag{3.19d}$$

$$\left\| \dot{\theta}(t) \right\|_{\infty} - \dot{\theta}_{max} \leq 0, \quad t \in [t_0, t_f], \tag{3.19e}$$

where $\dot{H}(\theta(t))_{update}$ is given by (3.18).

3.5 Penalty Formulation

To reduce the computational burden and enable real-time calculation, I reformulate the simplified optimal control problem of (3.19) into a penalty formulation to allow a trade-off between singularity avoidance and tracking accuracy. This ensures that the satellite will not reach a state where it cannot find any feasible solution to (3.19). The variables which are added to create the penalty formulation are the same as those used by Paradiso [23].

The first constraint is from the original OCP and comes from the physical capabilities of the satellite. The CMGs have a maximum rotation rate, and to measure the commanded gimbal rates which exceed this limit, I define

$$\dot{\theta}(t)_{over} = \max\left\{\left\|\dot{\theta}(t)\right\|_{\infty} - \dot{\theta}_{max},\ 0\right\}.$$

The second constraint was not in the original OCP and was added to the simplified OCP of (3.19) because of the new mission requirement to follow $\dot{H}(t)_{desired}$; so the new constraint is

$$\dot{H}(\theta(t))_{resid} = \left\|\dot{H}(\theta(t)) - \dot{H}(t)_{desired}\right\|_2^2.$$

The continuous form of the penalty formulation of the optimal control program, which will be used in discrete form in this thesis, is

$$\min_{\theta,u}\Bigg[W_1\int_{t_0}^{t_f}\frac{1}{m(\theta(t))}dt - W_2\min_{t\in[0,t_f]} m(\theta(t)) + W_3\int_{t_0}^{t_f}\dot{H}(\theta(t))_{resid}dt + W_4\int_{t_0}^{t_f}\dot{\theta}(t)_{over}dt\Bigg] \tag{3.20a}$$

$$\text{s.t.}\quad \dot{\theta}(t) = J(\theta(t))^{\dagger}\dot{H}(\theta(t))_{update} + u(t)\dot{\theta}_{null}(t), \quad t\in[t_0,t_f] \tag{3.20b}$$

$$\theta(t_0) - \theta_0 = 0, \tag{3.20c}$$

where $\dot{H}(\theta(t))_{update}$ is given by (3.18).

I note that although the original system dynamics of (3.3) still govern the satellite, knowing the values of the original state variables $\omega(t)$ and $q(t)$ does not affect the cost function or the inverse kinematics and so they are no longer necessary to the solution of the OCP. Therefore, calculation of $\omega(t)$ and $q(t)$ and can be accomplished after the algorithm has selected a trajectory, using the known values of $\omega(t_0)$ and $q(t_0)$.

Although the loss of a degree of freedom because of the mission path constraint $\dot{H}(\theta(t)) = \dot{H}(t)_{desired}$ will result in less optimal trajectories than could be found using the original OCP, the assumption is the first step in enabling autonomous trajectory improvement by satellites with limited computational capability.

In section 5 this continuous version of the optimal control problem will be changed to discrete form to allow for a discrete optimization over a limited search space, to reduce the computational complexity further and enable real-time computation by the satellite.

Chapter 4

Comparison to Other Methods

This chapter supplements the literature review of section 2.3, where the existing research in the CMG field was already discussed in high level terms. With the mathematical problem description and variable definitions laid out in chapter 3, it is now possible to describe precisely the differences in problem formulation of each method. Using the technical formulations, this section will clearly define the differences between the existing solution methods and the reasons for choosing the method developed in this thesis.

In section 4.1 I present an alternative measure of singularity to compare to equation (3.9). In section 4.2 I present several variations of the pseudoinverse, including the one which I will use in this thesis. Finally, in sections 4.3 and 4.4, I discuss the specific mathematical formulations of the solution methods discussed in section 2.3.

4.1 Singularity Measure

The measure of singularity employed throughout the greater quantity of literature in CMG research is that defined in (3.9). That is the singularity measure employed by this thesis to enable comparison between the results of my method and those of other methods.

Recently, Ford and Hall proposed a new measure of singularity using the smallest singular value S_{33} from the singular value decomposition of the Jacobian. Like the singularity measure of equation (3.9), Ford and Hall's singularity measure is not dependent on the size of the physical system, and is defined as

$$\sigma_{33} = \sqrt{\frac{3}{4}} \frac{S_{33}}{h_{CMG}}. \tag{4.1}$$

They noted that this singularity measure is bounded between 0 and 1, unlike the traditional measure, and they claim that it tends to create smoother profiles when applied to the singularity avoidance inverse scheme they developed [8].

It is worth noting that using Ford and Hall's pseudoinverse variant of equation (4.5) in conjunction with their singularity measure (4.1) would reduce the computational requirements of solving the problem since the singular value decomposition is only computed once and the determinant does not need to be computed as in (3.9).

However, the traditional measure of singularity is employed in this thesis, in order to directly compare the results of my method with the published results of other methods.

4.2 Pseudoinverse Variations

The variant of the pseudoinverse employed in this algorithm is from the Singular Direction Avoidance (SDA) method developed in [8] and defined in equation (4.5). This section explores the development of this inverse and the reason for choosing this variant above the others. The basic pseudoinverse is the Moore-Penrose (M-P) pseudoinverse which is given by equation (3.14). Unfortunately, when used in the inverse kinematics equation 3.16 the M-P inverse commands infinite gimbal rates. Variations of the M-P inverse are designed to be "singularity-robust", meaning that they still yield a finite value when the Jacobian is close to singularity.

The SDA method is derived from the more common variant of the pseudoinverse developed for robotics by [19] and known in the CMG literature as the Singularity Robust or SR-pseudoinverse described in [1] and given by

$$J^{\dagger,\mathrm{SR}} = J^T\Big(JJ^T + \rho\mathbf{1}_3\Big)^{-1}, \tag{4.2}$$

where $\mathbf{1}_3$ is the identity matrix and

$$\rho = \rho_0 e^{-\mu_0 m(\theta(t))}, \tag{4.3}$$

where ρ_0 and μ_0 are positive constants and $m(\theta(t))$ is defined in equation (3.9). Using the singular value decomposition of the Jacobian

$$J = USV^T,$$

where

$$S = \begin{bmatrix} S_{11} & 0 & 0 & 0 \\ 0 & S_{22} & 0 & 0 \\ 0 & 0 & S_{33} & 0 \end{bmatrix},$$

we find

$$J^{\dagger,\mathrm{SR}} = V \begin{bmatrix} \frac{S_{11}}{S_{11}^2+\rho} & 0 & 0 \\ 0 & \frac{S_{22}}{S_{22}^2+\rho} & 0 \\ 0 & 0 & \frac{S_{33}}{S_{33}^2+\rho} \\ 0 & 0 & 0 \end{bmatrix} U^T. \tag{4.4}$$

The scalar ρ increases exponentially in proportion to the measure of CMG gain, $m(\theta(t))$. This variant of the pseudoinverse adds a small torque error to all three columns of JJ^T to prevent the pseudoinverse from approaching infinity. However, in [16] and [8] the authors showed it is only necessary to add torque error in the singular direction. Therefore, the SR-inverse adds unnecessary torque error which is undesirable for precision pointing missions. The SDA pseudoinverse used in this thesis addresses this issue.

The only torque error necessary to avoid the singular state as CMG gain $m(\theta(t))$ decreases can be added solely in the direction associated with the smallest singular value. In this way, the singular direction is avoided and total torque error is kept small. If the singular values are ordered $S_{11} \geq S_{22} \geq S_{33}$, then the SDA pseudoinverse is

given by

$$J^{\dagger,SDA} = V \begin{bmatrix} \frac{1}{S_{11}} & 0 & 0 \\ 0 & \frac{1}{S_{22}} & 0 \\ 0 & 0 & \frac{S_{33}}{S_{33}^2+\alpha} \\ 0 & 0 & 0 \end{bmatrix} U^T, \tag{4.5}$$

where $\alpha = \alpha_0 e^{-\mu m(\theta(t))}$, and α_0, and μ are positive constants and $m(\theta(t))$ is defined in equation (3.9) as a measure of closeness to singularity. The value of α is negligible when $m(\theta(t))$ is sufficiently large, meaning that the system is sufficiently far from singularity. When α is negligible, $J^{\dagger,SDA} \cong J^{\dagger}$ because no torque error is necessary to ensure the pseudo-invertibility of the Jacobian. When $m(\theta(t))$ is small and the system is close to singularity, torque error is added only in the direction of the singular vector corresponding to the near-singular singular value.

In recent years, researchers such as Bong Wie have proposed other variants of the pseudoinverse which are more successful than SDA at using torque error to escape singularities [27]. However, the primary goal of my algorithm is to avoid singularities using null motion rather than escaping them using torque error, and Bong Wie's method is a poor fit for that goal.

Because it is only necessary to add torque error in the singular direction, the SDA method of computing the inverse induces less total torque error than the SR-method and other previous methods [16], [8]. For that reason, the SDA method will be used throughout this thesis to compute the particular solution of the inverse kinematics given in equation (3.16).

4.3 Local Methods

The variants of the pseudoinverse described in section 4.2 are one type of local method employed to steer the gimbals and escape the problems associated with singular configurations of the CMGs. Another type of method to avoid singular configurations is to use null motion to avoid approaching the singular configuration at all. These local methods do not use any form of the optimal control problem presented in (3.11) but instead use only variations of the inverse kinematics of (3.16) to make locally optimal decisions and attempt to keep the CMG gain from approaching 0. Bedrossian discussed one such singularity avoidance method known as the Local Gradient (LG) method [1]. This method uses null motion to keep the singularity measure at a local optimum. However, as Bedrossian pointed out, the locally optimal solutions tend to drift slowly towards singularity. The LG method alone is insufficient to avoid singularities [1].

Leve and Fitz-Coy developed the most recent local solver to be added to the field of CMG research. It is a combination of a local escape method, the SDA inverse, and a local avoidance method, the LG method [16]. Their reasons for choosing to combine these two methods are not identified, but the use of the SDA inverse can be explained by research in the field indicating that it minimizes unnecessary torque error [8]. However, both Bedrossian and Paradiso pointed out that the LG method tends to drift towards singularity [1] [22]. Although Leve and Fitz-Coy applied metrics to reduce the amount of LG method applied near elliptic singularity, they do not explain

their use of the LG method near hyperbolic singularity.

Leve and Fitz-Coy's method is a recent development in the inverse kinematics [16]. Building on the development of the SDA method by Ford and Hall [8] and and using the derivation of the classification matrix Q developed by Margulies and Aubrun [17] and shown in Appendix B, they hypothesized that it was only necessary to apply torque error only when the configuration was in an elliptic singularity. Since hyperbolic singularity can be escaped using null motion, they proposed that the total torque error can be reduced by developing an inverse kinematic equation which takes into account the type of singularity being approached. Using the matrix Q given in B.9, which has a positive determinant at elliptic singularities and a negative or zero determinant at a hyperbolic singularity, they defined an alternative scaling constant to control the torque error to be applied in the SDA inverse, given by

$$\bar{\alpha} = |Q_0 - det(Q)| \tag{4.6}$$

$$\alpha(\theta(t)) = \alpha_0 e^{-a\bar{\alpha}} e^{-\mu_1 m(\theta(t))}, \tag{4.7}$$

where a, Q_0, and μ_1 are positive constants. Similarly, they define a scaling constant to control the null motion to be applied in the inverse kinematics, given by

$$\bar{\beta} = \frac{1}{|Q_0 - det(Q)|} \tag{4.8}$$

$$\beta(\theta(t)) = \beta_0 e^{-b\bar{\beta}} e^{-\mu_2 m(\theta(t))}, \tag{4.9}$$

where b, and μ_2 are positive constants.

Both equations are similar to equation (4.3) with the exception of the first ex-

ponential terms. The first terms of equations (4.7) and (4.9) scale according to the type of singularity which the CMGs are approaching. Whereas the value of equation (4.3) is large whenever the system is close to any singularity, $\alpha(\theta(t))$ is significant only when the system is close to elliptic singularity, and $\beta(\theta(t))$ is significant only when the system is close to hyperbolic singularity. Leve and Fitz-Coy argued that adding torque error near hyperbolic singularity is unnecessary and undermines tracking precision, and that null motion near elliptic singularity is useless, but Bedrossian showed in [1] that torque error can improve the effect of null motion in *near*-singular states by slowing the response of the system. Additionally, Leve and Fitz-Coy did not clearly define the value of the scalar Q_0, which makes their method difficult to reproduce. It is not clear why $det(Q)$ is bounded, and Leve and Fitz-Coy specified only that Q_0 should be "on the same order of magnitude of $det(Q)$ only greater" [16, p. 1205]. An improper selection of Q_0 would undermine the desired effects of $\alpha(\theta(t))$ and $\beta(\theta(t))$.

Leve and Fitz-Coy use the scaling constants defined in equations (4.7) and (4.9) to create their unique inverse kinematics, given by

$$\dot{\theta}(t) = J(\theta(t))^{\dagger,\alpha(\theta(t))}\dot{H}(\theta(t)) + \beta(\theta(t))\left[\mathbf{1}_4 - J(\theta(t))^{+}J(\theta(t))\right]\dot{\theta}_{null}. \tag{4.10}$$

The authors use the Local Gradient method to scale the null motion near hyperbolic singularity when $\beta(\theta(t))$ is large. The reason for the choice is not explained, and Bedrossian's research [1] indicates that the Local Gradient method is frequently drawn into elliptic singularity when applied as a local method. Bedrossian also noted that

while near but not at elliptic singularity, the biggest effect of torque error is to "slow the system response...allowing more time for null motion to be applied" [1, p. 129]. Therefore, Leve and Fitz-Coy's method, which suppresses null motion while applying torque error, will be improved upon in this thesis by allowing null motion to be applied at any time, as long as it results in an improved overall trajectory.

Paradiso noted in [22] that the Local Gradient method keeps the system on a local ridge which appears to be locally optimal. However, the local ridges frequently end in singularity. For this reason, Paradiso hypothesized and successfully showed in [22] and [23] that considering a variety of choices for null motion, including choices which seem at a local level to make the CMG gain move closer to singularity, results in a more globally optimal solution. For this reason, this thesis will improve existing research by creating a hybrid approach similar to that of Leve and Fitz-Coy which combines a simple escape method (SDA) with the search-based method of choosing null motion developed by Paradiso. By combining a local escape method with a search method, this thesis bridges the gap between local and global methods by creating a predictive "look-ahead" method.

4.4 Global Methods

Local methods do not make use of an optimal control problem to optimize the trajectory, but instead make local decisions which seem optimal based on local information. By a global method, I mean a method which takes advantage of information about

the result of local decisions on a the trajectory in the future. These include global methods which solve the Optimal Control Problem (OCP), and search methods such as that of Paradiso. For example, a two-point boundary value solver is a global solver which would solve the complete OCP defined in (3.11), with or without path constraints $\dot{H}(t)_{desired} = \dot{H}(\theta(t))$. Usually, the control of the system is $\dot{\theta}(t)$. When the path is not constrained, the extra degree of freedom means that other objective functions can be used in place of 3.11a, such as optimizing t_f.

Whereas in the solution method proposed by this thesis I explicitly choose the values of the null scalar in order to control the application of null motion of the system, in a two-point boundary value solver null motion is chosen implicitly by using $\dot{\theta}(t)$ as the control variable. Although this method successfully avoids singularities, the cost of calculation means that it is still impossible for a satellite to autonomously compute the solution to the two-point boundary value problem fast enough to correct for disturbances in real time. However, solutions such as this are calculated offline on the ground and can be successfully used to control a satellite in the absence of significant disturbances by uploading the pre-calculated trajectory to the satellite system. The robust capabilities of software packages such as DIDO easily solve the two-point boundary value problem to avoid singularities.

However, the importance of developing a method which can be employed in real-time onboard the satellite and enable it to make intelligent decisions based on predictive trajectory information means that methods such as that less computationally

complex methods such as that proposed in this thesis will be in high demand until satellites become technologically capable of solving the two-point boundary value problem in real-time.

In Paradiso's original exploration of the directed search as a method of intelligently choosing null motion to avoid and minimize the negative effects of singularities in [22] and [23], he used system dynamics which differed from those of this thesis only in the choice of pseudoinverse, using the inverse kinematic equation

$$\dot{\theta}(t) = J(\theta(t))^{\dagger,SR}\dot{H}(\theta(t)) + u(t)\dot{\theta}_{null}(t), \quad t \in [t_0, t_f]. \tag{4.11}$$

Paradiso employed the so-called Singularity Robust inverse of equation (4.2). Paradiso pioneered the idea of discretizing the system dynamics and searching over a discrete set of potential trajectories by identifying 'decision nodes' to limit the potential choices of null motion [22]. He chose to explore a search space using limited choices for the null motion scalar of $[-1, 0, 1]$. His search method is the most successful example of a global approach which is capable of finding a near-optimal trajectory in a global sense, and his success with it in 1991 using a Macintosh II and techniques which have since been improved upon by other researchers is the reason why it is the foundation of the method proposed in this thesis.

This thesis proposes a search method using the improved SDA inverse of (4.5) in the heritage of Paradiso's discrete search which requires a discrete problem formulation.

Chapter 5

Discretization

To improve upon existing methods and enable real-time look ahead guidance for the satellites, I developed a streamlined form of the discrete search-based approach of Paradiso by removing unnecessary terms from the cost function and eliminating the use of multiple cost functions [22]. In section 5.1, I develop the discrete graphical framework for the search space and define it using graph theory terminology. Then in sections 5.2 and 5.3, I develop the discrete system dynamics to allow the trajectory selected by a discrete search to closely predict the behavior of a real satellite in a continuous system. Additionally, I define the discrete penalty formulation of the optimal control problem which my algorithm will use to choose between investigated trajectories; this is a simplified form of the cost function developed by Paradiso [22].

With the elements of a discrete graphical search in place, it is natural to wonder why Dynamic Programming or Shortest Path Planning would not be far simpler

methods of choosing an optimal trajectory; I address these concerns briefly in section 5.4 and then define my new search-based algorithm in section 5.5.

In chapter 6, I develop a real-time framework to apply my search algorithm and numerically evaluate the feasibility and reliability of my method.

5.1 Discrete Search Space

First it is necessary to develop a discrete tree based on the CMG problem described in section 3 over which the algorithm will search for a near-optimal trajectory. Each node on the tree represents a state of the system and the path from the root of the tree to that node represents the trajectory taken to reach that state. At a depth of 3, the search tree is shown graphically in figure 5.1.

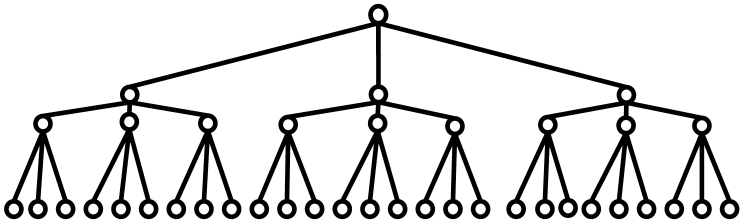


Figure 5.1: The topmost node represents the starting conditions of the system, $\theta(t_0)$. Nodes in line horizontally represent potential states of the system at time t_k. A path from $\theta(t_0)$ to another node represents a potential control sequence which will yield the trajectory necessary to reach that node.

At the start of the satellite maneuver, the values of the state $\theta(t_0)$ are known. I assign $\theta(t_0)$ be the first node of the graph T_3. To limit the computational complexity of

the problem, I consider three choices for values of the discrete control; for convenience, I choose $u(t_k) \in \{-1,\ 0,\ 1\}$. Then using the discrete differential equations in equation (5.3), I first calculate $\dot{\theta}(t_0)$ for each value of $u(t_0) \in \{-1,\ 0,\ 1\}$ and use a discrete integration method to calculate $\theta(t_1)$ for each value of $u(t_0)$.

Therefore if $\theta(t_0)$ is the parent node, then there are three children nodes which represent $\theta(t_1)$. Each of the three nodes corresponding to $\theta(t_1)$ also has three child nodes. Therefore the number of nodes expands exponentially with increasing values of k. The only nodes which do not have child nodes are the nodes which correspond to values of $\theta(t_f)$.

At $\theta(t_N)$ we have 3^N grandchild nodes of the original parent node $\theta(t_0)$. In graph theory terminology I define $\theta(t_0)$ to be the root of a T_3 tree. Then the grandchild nodes corresponding to $\theta(t_f)$ are the leaves of T_3. Each leaf represents a potential final state of a satisfactory satellite trajectory on $[t_0, t_N]$.

By adapting the definition of a perfect binary tree in [14], I define T_3 as a perfect tertiary tree: a rooted tree where each node other than the leaves has exactly three children corresponding to $\{-1,\ 0,\ 1\}$, and all the leaves are the same distance from the root. Then, as Diestel did for infinite binary trees in [4, p. 213], I note that each vertex of the satellite search tree T_3 can be uniquely identified by the finite $[-1,\ 0,\ 1]$ sequence identifying its parentage. Then the set of vertices of a finite perfect tertiary tree with distance N between the root and each leaf corresponds exactly to the $\{-1,\ 0,\ 1\}$ sequences of length N or less, and the root of the tree

is identified by the empty set [4, p. 212]. This is depicted graphically in figure 5.2. The $\{-1,\ 0,\ 1\}$ sequences of depth 1 correspond to potential values of $\theta(t_1)$, those of depth 2 correspond to potential values of $\theta(t_2)$, while the $\{-1,\ 0,\ 1\}$ sequences of length N correspond to potential final states $\theta(t_f)$ of the satellite system after a complete trajectory. These sequences correspond exactly to the control vector $\mathbf{u}_N = [u(t_0),\ u(t_1),\ \ldots\ u(t_N)]$ for each satellite trajectory, and so can be used to identify the discrete control sequence required to follow that trajectory.

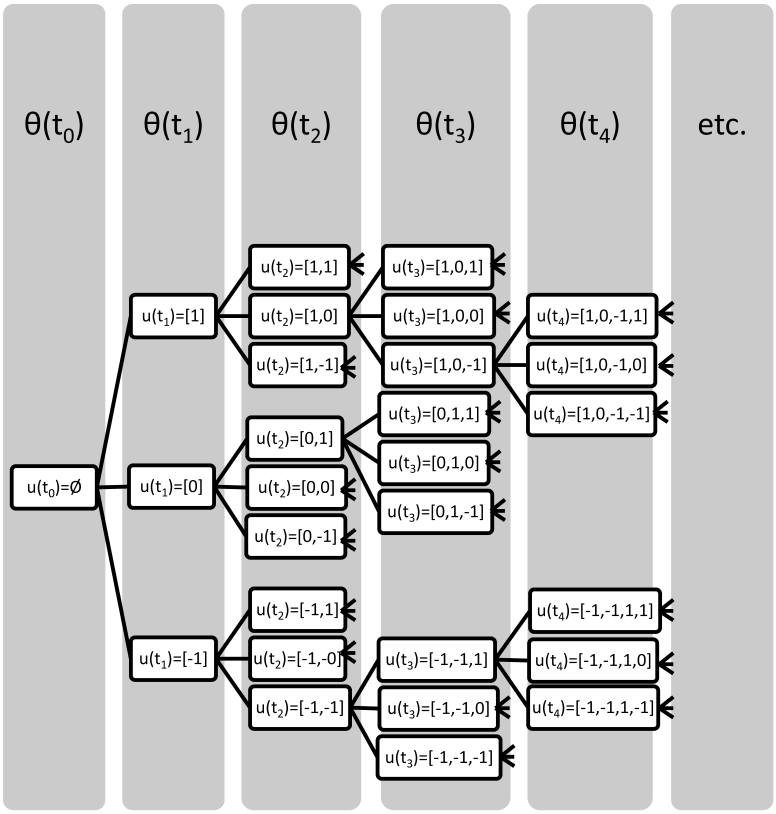


Figure 5.2: The rooted perfect tertiary tree: each node is a partial trajectory with a unique $\{-1,\ 0,\ 1\}$ control sequence required to reach it. Adapted from [4, p. 213].

Next I discretize the continuous system dynamics of the penalty formulation OCP (3.20) to show how to calculate the state of the satellite at each node. The objective function of the discrete OCP can be used to assign a cost value to each node of T_3.

5.2 Time Discretization

Recall that the mission of the satellite in the CMG problem is to follow a particular body rate and acceleration profile $\omega_{desired}(t)$ and $\dot{\omega}_{desired}(t)$ defined using the methods of Appendix A. Because the desired maneuver is preselected, t_0 and t_f are defined.

I partition the time interval into two meshes: a rough mesh P_k with $N+1$ elements and a fine mesh P_m with $n+1$ elements such that $P_k \subset P_m$. The rough mesh will be identified as the *decision nodes* of the graph. The fine mesh will be used as a discrete time vector.

Although the null scaling constant changes only on the rough mesh, the null vector and Jacobian are instantaneous quantities and so must be updated on the fine mesh. Suppose $\dot{\theta}_{null}(t_m) \in N\Big(J(\theta(t_m))\Big)$ for some $t_m \in P_m$ where N is the null space of the Jacobian. Let $\theta(t_{m+1}) = \dot{\theta}(t_m)\Delta t + \theta(t_m)$. Then in general, $J(\theta(t_m)) \neq J(\theta(t_{m+1}))$. However, $\lim_{\Delta t \to 0} \theta(t_{m+1}) = \theta(t_m)$, and equation (3.8) shows that the Jacobian is a matrix of continuous functions in $\theta(t)$ so $\lim_{\theta \to \theta(t_m)} J(\theta) = J(\theta(t_m))$. Then by the theorem of composite limits [15, p. 49],

$$\lim_{\Delta t \to 0} J(\theta(t_{m+1})) = J(\theta(t_m)),$$

and so we can show

$$\lim_{\Delta t \to 0} N\Big(J(\theta(t_{m+1}))\Big) = N\Big(J(\theta(t_m))\Big).$$

Therefore, for small enough Δt we can approximate continuous null motion using discrete null motion over Δt. This is the reason the use of the fine mesh P_m is so important for accuracy in discretizing the system dynamics: it is important to continuously update $J(\theta(t_m))$ and $\dot{\theta}_{null}(t_m)$.

The rough mesh will be used as 'decision nodes' for the algorithm. As defined in section 3.5, the control for the continuous system is $u(t) \in \mathbb{R}$. In the discrete system, a discrete array of choices is chosen for $u(t_k)$, and $u(t_k)$ is constant on the fine mesh for the interval in between nodes of the rough mesh. That is, $u(t_m) = u(t_k)$ for all $t_m \in [t_k, t_{k+1})$. Throughout this thesis, I only consider $u(t_k) \in [-1,\ 0,\ 1]$ for all $t_k \in [t_0,\ t_1,\ \ldots\ t_N]$.

The number of elements of P_k and the number of choices for $u(t_k)$ together determine the computational complexity of the algorithm. If $u(t_j) \in [-1,\ 0,\ 1]$ and the partition P_k has $N+1$ elements, then the number of leaves of T_3 is 3^N. So as N increases, the size and complexity of T_3 increases exponentially. To integrate on a finer mesh than P_k, I need P_m to have at least ten times as many elements as P_k. Therefore, to limit the computational requirements of this algorithm, it is impossible for every element of P_m to be a decision node. Computational complexity is also the reason why the options for $u(t_k)$ are limited to three.

5.3 Discrete System Dynamics

Using the selected time discretization P_m and decision node discretization $P_k \subset P_m$, I create the discrete form of the optimal control problem (3.20).

First I calculate the discrete form of the desired rotation profile from $\mathbf{e}_0$ to $\mathbf{e}_f$, $\mathbf{e}(t_k)$ for $t_k \in P_k$. Then, using the methods of Appendix A I calculate the desired body rate and acceleration profile $\dot{\omega}(t_k)_{desired}$ and $\omega(t_k)_{desired}$. To mimic the control behavior of the actual satellite, I assume that $\dot{\omega}(t_m)_{desired} = \dot{\omega}(t_k)_{desired}$ for all $t_m \in [t_k, t_{k+1})$. Therefore, $\omega(t_m)_{desired}$ changes linearly for $t_m \in [t_k, t_{k+1})$.

Next using a discretized version of equation (3.12) and $H(\theta(t_0))$, I generate

$$\dot{H}(t_k)_{desired} = -\mathbf{I}\dot{\omega}(t_k)_{desired} - \Big(\omega(t_k)_{desired} \times \big(\mathbf{I}\omega(t_k)_{desired} + H(t_k)_{desired}\big)\Big). \quad (5.1)$$

As with $\dot{\omega}(t_k)_{desired}$, due to the mechanics of the satellite control, it is safe to assume that $\dot{H}(t_m)_{desired} = \dot{H}(t_k)_{desired}$ for all $t_m \in [t_k,\ t_{k+1})$. Similarly,

$$\dot{H}(\theta(t_m))_{update} = -\mathbf{I}\dot{\omega}(t_m)_{desired} - \Big(\omega(t_m)_{desired} \times \big(\mathbf{I}\omega(t_m)_{desired} + H(\theta(t_m))\big)\Big). \quad (5.2)$$

Then the discrete inverse kinematics are given by

$$\dot{\theta}(t_m) = J(\theta_m)^{\dagger}\dot{H}(t_m)_{update} - u(t_m)\dot{\theta}_{null}(t_m), \quad (5.3)$$

where $\dot{\theta}_{null}(t_m) \in N\Big(J(\theta(t_m))\Big)$ where $N\Big(J(\theta(t_m))\Big)$ is the null space of the Jacobian, and $u(t_m)$ is constant for the interval between decision nodes $t_m \in [t_k, t_{k+1})$. The reasons for using $\dot{H}(t_k)_{desired}$, $\dot{H}(t_m)_{update}$, and the inverse kinematics are explained in section 3.4.

For the penalty formulation, I define discrete variables to represent the constraints. First is the path constraint, given by

$$\dot{H}(t_m)_{resid} = \left\| \dot{H}(\theta(t_m)) - \dot{H}(t_m)_{desired} \right\|_2^2.$$

Second is the hardware constraint on the gimbal rates, given by

$$\dot{\theta}(t_m)_{over} = \max\left\{ \left\| \dot{\theta}(t_m) \right\|_\infty - \dot{\theta}_{max},\ 0 \right\}$$

Then the discrete optimal control problem is given by

$$\min_{\theta,u} \left[W_1 \sum_{i=0}^{n} \left(\frac{1}{m(\theta(t_i))} \right) - W_2 \min_{t_i \in P_m} m(\theta(t_i)) + W_3 \sum_{i=0}^{n} \dot{H}(t_i)_{resid} + W_4 \sum_{i=0}^{n} \dot{\theta}(t_i)_{over} \right] \tag{5.4a}$$

s.t.

$$\dot{\theta}(t_m) = J(\theta(t_m))^\dagger \dot{H}(\theta(t_m))_{update} + u(t_m)\dot{\theta}_{null}(t_m), \quad t_m \in P_m \tag{5.4b}$$

$$\theta(t_0) - \theta_0 = 0, \tag{5.4c}$$

where $\dot{H}(\theta(t_m))_{update}$ is given by (5.2).

In my algorithm I define $\theta(t_{m+1}) = \dot{\theta}(t_m)(t_{m+1} - t_m) + \theta(t_m)$ for computational efficiency. Then, given a sequence of null motion decisions $u(t_k)$ made at each decision node and held constant for $u(t_m)$ on $t_m \in [t_k, t_{k+1})$ I can calculate the states $\theta(t_m)$ for the entire trajectory. Then the body rate and attitude profiles can be computed in post-processing using the discrete forms of the differential equations (3.3) and (3.4),

given by

$$\dot{\omega}(t_m) = \mathbf{I}^{-1}\Big(-\omega(t_m) \times \Big(\mathbf{I}\omega(t_m) + H(\theta(t_m))\Big) - \dot{H}(\theta(t_m))\Big) \tag{5.5}$$

$$\dot{q}(t_m) = \frac{1}{2} T(q(t_m)) \left[0\ \omega(t_m)^T\right]^T. \tag{5.6}$$

The scalars $\mathbf{W} = [W_1,\ W_2,\ W_3,\ W_4] \in \mathbb{R}^4$ in equation (5.4a) are weights included to allow more flexibility in the cost function, when some requirements might be more flexible than others. In my algorithm I use the weights given in table 5.3; I numerically verified that my algorithm produces good trajectories using these scaling constants but I will explain why these weights make sense in order from greatest weight to least.

Scalar	Value:
W_1	3
W_2	20
W_3	2
W_4	100

Table 5.1: The objective function weights used throughout this thesis, chosen after numerical simulation to determine that they produce desirable trajectories efficiently.

The largest weight is $W_4 = 100$ because the fourth term of equation (5.4a) represents a hardware constraint. The CMGs are not physically capable of exceeding $\dot{\theta}_{max}$ and so a nonzero value of the fourth term means that the trajectory, as calculated,

is not feasible. When the satellite attempts to follow such a trajectory, it will be accomplished slower than anticipated.

The next largest weight is $W_2 = 20$ because the second term measures the closest distance to singularity which the trajectory approaches, and maintaining the distance of the system from singularity is the primary goal of this thesis' algorithm. It also affects the third term to some extent: so long as the system remains reasonably far from singularity, no torque is added to the system and so $\dot{H}(t_m)_{resid} = 0$ for all $t_m \in P_m$. Only when the system nears a singularity does the third term become important as tracking accuracy is traded in favor of singularity avoidance.

The second smallest weight is $W_1 = 3$, on the first term of equation (5.4a) which optimizes the average distance of the system from singularity. This term also affects the third term. If the system does approach singularity, less torque error is incurred if the first term is kept as high as possible because the time spent near singularity is minimized.

The smallest weight is $W_3 = 2$ because if the first and second terms are prioritized and the system is kept far from singularity, $\dot{H}(t_m)_{resid} = 0$. When the algorithm cannot find a trajectory which remains far from singularity, $\dot{H}(t_m)_{resid} \neq 0$ and tracking accuracy is balanced against singularity avoidance.

The objective function (5.4a) assigns a cost to each complete trajectory. However, it can also be used to assign a cost to each partial trajectory, represented as a node

$\theta(t_k)$ of the discrete search tree T_3. This cost function is

$$\begin{aligned} C_0^k =& W_1 \sum_{i=0}^{r} \left(\frac{1}{m(\theta(t_i))} \right) - W_2 \min_{t_0 \le t_i \le t_r} m(\theta(t_i)) \\ &+ W_3 \sum_{i=0}^{r} \dot{H}(t_i)_{resid} + W_4 \sum_{i=0}^{r} \dot{\theta}(t_i)_{over}, \end{aligned} \tag{5.7}$$

where r is the index of the node on the fine mesh $t_r \in P_m$ corresponding to the decision node $t_k \in P_k$ where $\theta(t_k)$ is defined, such that $t_r = t_k$. Equation (5.7) gives a trajectory-dependent cost value to each node of the tree. Because the second term is not a summation but a minimum over the trajectory, it cannot be formulated as a trajectory-independent cost for each node.

With a discrete problem formulation, is natural to wonder whether it would be possible to use Dynamic Programming or Shortest Path Planning to solve it. This will be addressed in section 5.4.

5.4 General Solution Approaches

Having developed a discrete problem formulation, there are a pair of textbook techniques for solving discrete problems which at first glance may seem like a natural fit for solving the CMG optimization problem. Before presenting the method derived in this thesis, I will first explain why the well-developed techniques of Dynamic Programming and Shortest Path Planning cannot be applied to this problem.

5.4.1 Dynamic Programming

Dynamic Programming, in its most basic form, is a method of "partial enumeration: a method of finding the best alternative without listing all possible alternatives" [6]. At first glance, this seems like a practical way to solve an exponential tree search without listing all of the possible trajectories.

According to Bellman's principle of optimality [25, p. 362], given a globally optimal trajectory which minimizes (5.4a),

$$\boldsymbol{\theta}^\star = [\theta^\star(t_0),\ \theta^\star(t_1),\ \dots\ \theta^\star(t_n)] \in \mathbb{R}^{4\times(n+1)},$$

then for any of time $t_m \in P_m$, the same trajectory $\boldsymbol{\theta}^\star$ is optimal from t_m to t_n. That is,

$$\boldsymbol{\theta}^\star = [\theta^\star(t_m),\ \theta^\star(t_{m+1}), \dots \theta^\star(t_n)] \in \mathbb{R}^{4\times(n-m+1)},$$

is the optimal trajectory which minimizes $C_m^n(\boldsymbol{\theta})$ starting from that node $\theta(t_m)$. This is intuitive when thinking about optimal trajectories on trees. If another branch starting at $\theta(t_m)$ were optimal, then it would have been on the optimal branch starting at $\theta(t_0)$. Suppose there exists some trajectory $\hat{\boldsymbol{\theta}}$ such that $C_m^n(\hat{\theta}) < C_m^n(\boldsymbol{\theta}^\star)$. Then

$$C_0^{m-1}(\boldsymbol{\theta}^\star) + C_m^n(\hat{\boldsymbol{\theta}}) < C_0^{m-1}(\boldsymbol{\theta}^\star) + C_m^n(\boldsymbol{\theta}^\star)$$

This contradicts the assumption that $\boldsymbol{\theta}^\star$ was optimal.

Since that is true for the CMG problem, dynamic programming seeks to define a recursive relationship to minimize the effort required to solve the problem.

The primary reason I cannot use dynamic programming to solve the discrete CMG problem is that using the objective function of (5.7) as a cost function to assign a cost to each node of T_3, I cannot define a trajectory-independent cost which could assign a cost to each node independently of the trajectory taken to reach it. By removing the problematic term $\min_{t_i} m(\theta(t_k))$ from the cost equation (ignoring for now its critical importance to the CMG problem), equation (5.7) can be formulated as a seemingly trajectory-independent node cost,

$$\hat{f}(\theta(t_k)) = W_1 \frac{1}{m(\theta(t_k))} + W_3 \dot{H}(t_k)_{resid} + W_4 \dot{\theta}(t_k)_{over}, \tag{5.8}$$

where $\theta(t_k)$ is a node of depth k from the root of the tree. Recall that there are 3^k nodes at each depth of the tree (t_0 has 1 node, t_1 has 3, etc.).

Let $C_k^j(\boldsymbol{\theta}(t_k))$ for $t \in [t_k,\ t_{k+1},\ \ldots\ t_j] \subseteq P_k$ be the cost of the optimal trajectory between $\theta(t_k)$ and any of the 3^{j-k} nodes of the tree $\theta(t_j)$ which are accessible from $\theta(t_k)$. Then $C_k^j(\boldsymbol{\theta}(t_k))$ can be written as a summation of trajectory independent costs of individual nodes on the optimal trajectory,

$$C_k^j(\boldsymbol{\theta}(t_k)) = f(\theta(t_j)) + \sum_{i=k}^{j-1} f(\theta(t_i)). \tag{5.9}$$

Then in the dynamic programming formulation, I define a recursive relationship

$$C_k^N(\boldsymbol{\theta}(t_k)) = \min_{j \in [1,\ 2,\ 3]} \left[f(\theta(t_k)) + C_{k+1,j}^N(\boldsymbol{\theta}(t_{k+1,j}) \right], \tag{5.10}$$

where $\boldsymbol{\theta}(t_{k+1,j})$ is the j^{th} child of node $\boldsymbol{\theta}(t_k)$. This would be the formulation used to apply dynamic programming *if* it were possible to formulate a trajectory independent

cost function like (5.8) instead of the trajectory dependent cost function I use in my search-based optimization method [6].

However, having developed the recursive relationship it is now possible to see why dynamical programming is not a good solution approach for the CMG problem. The first step in solving the complete optimization problem $C_0^N(\boldsymbol{\theta}(t_0))$ is to solve the last step, $C_{N-1}^N(\boldsymbol{\theta}(t_{N-1}) = \min_{1<j<3^N} f(\theta(t_{N,j}))$. This requires using (5.3) to calculate every one of 3^N trajectories in order to know the states of $\theta(t_N)$ at each one of the leaves of the tree since the state $\theta(t_N)$ cannot be calculated separately from the trajectory required to reach it. So the goal of partial enumeration which is the reason to employ dynamical programming is unattainable since any attempt to use it leads to "accidental" complete enumeration, which is computationally prohibitive except for small values of N.

The trajectory-dependent nature of the cost function means that it is impossible to compute the cost of a node without knowing the costs of all its ancestors on the trajectory. So dynamical programming cannot be employed effectively. In this thesis, the first attempt to solve the problem did employ a recursive relationship much like a dynamical programming formulation. I quickly discovered the inefficiency of the problem as it tried to calculate cost values for exponential trajectories on the tree! This issue was already addressed by Paradiso in [22] in his review of the A* search method, which is a complete enumeration search method and is what any recursive or dynamic programming approach degenerates into.

5.4.2 Shortest Path Planning

For the CMG T_3 tree, a shortest path planning problem would seek the path between the root of the tree $\theta(t_0)$ and any leaf $\theta(t_N)$ such that the sum of the costs of the edges in p is minimal. Alternatively, all of the leaves can be assigned a single dummy child node 'END' as in figure 5.3; then the problem seeks the path from $\theta(t_0)$ to 'END' such that the cost of the path is minimal. The cost to travel from a leaf node of T_3

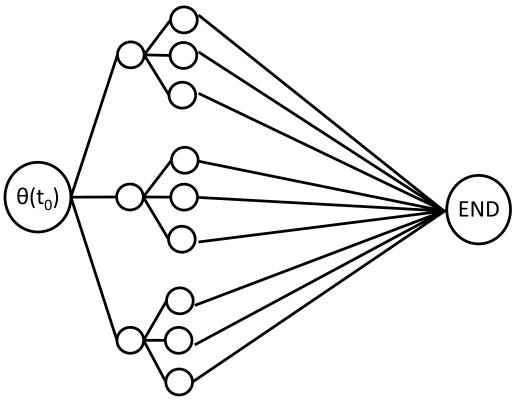


Figure 5.3: The CMG problem as a shortest path planning problem, with a dummy 'END' node as the child of every leaf. Then the problem is to find the path from $\theta(t_0)$ to the 'END' node with the smallest cost.

to the 'END' node is 0. The cost of each node $\theta(t_k)$ is given by equation (5.7).

Algorithms exist to solve this problem very efficiently for graphs where the costs of each edge is known in advance. For the satellite CMG problem, the value of the cost function at each node is not known until a trajectory is tested. With 3^N trajectories available to the discrete search, the computational cost of calculating the cost function value at each node and storing the data is prohibitive. It is also true that many of

these trajectories will be sub-optimal, so it is a computational waste to explore them. In the CMG problem formulation, the value of the cost function at each node is found by exploring the tree, at the expense of computation or exploration cost. With high computational resources, complete information about the tree could be gathered and then one of the many well-developed algorithms for solving the Shortest Path Problem could be applied. However, this computational power is not available on a satellite. I must reduce the computational requirements of the algorithm.

The specific subset of the shortest path problem which is most like the CMG problem is the so-called 'Canadian Traveler Problem', so named because in Canada in the winter, drivers are forced to find their way to a destination when information is not available about which roads are blocked due to snow [14]. Information is gathered by exploring the open roads, but it takes time to gather complete information about the cost (closed or open) of each road in the city. The Canadian Traveler Problem deals with finding an efficient trade-off between the cost of gathering information and the cost of the final route chosen to arrive at the destination.

Although in general the optimal strategies for solving general Canadian Traveler Problems are not known, Karger and Nikolova (2008) derived an exact solution for a Canadian Traveler Problem for perfect binary trees with costs of 0 or 1 according to a Bernoulli distribution [14] which is somewhat similar to the method proposed by this thesis to explore the discrete selection of trajectories whose costs are not known in advance.

In the Canadian Traveler Problem with Bernoulli distribution, the optimal policy to balance exploration against the final cost of the chosen trajectory is as follows. First, all available 0-cost edges are explored. This is free exploration of the graph, and leads to maximum information gathering with minimal cost commitment. Then all unexplored edges have cost 1. The node which is closest to the goal node is selected, the cost 1 edge is traversed, and the search restarts by exploring all 0-cost edges available. The reason this policy is optimal is because, with a Bernoulli distribution, the expected value of the unknown portion of the trajectory is a simple function of the distance from the node of interest to the goal node. The node which is closest to the goal node has the smallest expected value for the remainder of its trajectory; so the optimal choice is to proceed from that node and ignore exploration of the rest of the graph.

Although there is no expected distribution of costs of the edges in the CMG problem, and there are no 'free' or 0 cost edges since every edge has the same computational cost to transverse, still the "optimal policy" which Karger and Nikolova proposed for finding a near-optimal route while minimizing exploration cost [14] is somewhat reminiscent of the search method proposed by this thesis. So although an optimal policy has not been proved for Canadian Traveler Problem matching the CMG problem of this thesis, the optimal policy on perfect binary trees with (0,1) Bernoulli distribution of edge costs does have some bearing on the algorithm developed in this thesis to seek out near-optimal trajectories using a discrete search.

5.5 Directed Search Method

The algorithm developed in this thesis is a somewhat simpler form of the "guided depth–first search" developed by Paradiso in [22]. This thesis reduces the complexity of his method to enable real-time application. At the start of the algorithm, the values describing the state of the satellite system at the root node of the discretized tree are known.

The search method has two parts and will be explained in two subsections. To begin the algorithm, I perform some low-cost exploration to gather information about the tree. Next, I repeatedly perform greedy-depth first searches to investigate promising trajectories on the search tree.

In the third subsection I present the results of using the search algorithm in its basic form to select rotation trajectories for the satellite.

5.5.1 Initial Graph Exploration

The first step of the search method is to perform some low-cost exploration to gather information about many potential trajectories to exploit in the directed search.

Specifically, I investigate three basic trajectories starting from the root node: see figure 5.4. Using the discrete dynamics given in section 5.3, I calculate the value of $\dot{\theta}(t_0)$ for three values of $u(t_0) \in \{-1\ 0\ 1\}$ and use the forward Euler method to calculate the values of $\theta(t_1)$ at the three child nodes of the root node.

The first trajectory I investigate uses all positive null motion. So after each of the

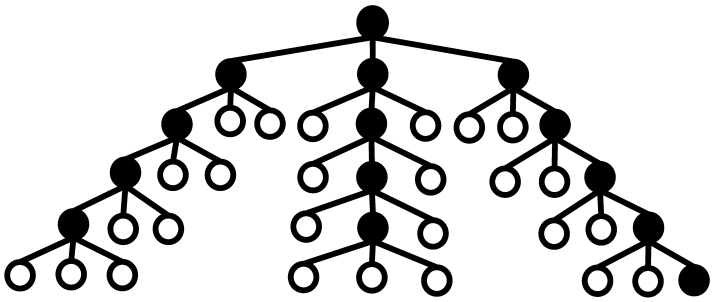


Figure 5.4: Explore three basic trajectories to open a database of potential nodes: one with all positive null motion (right), one with no null motion (center), and one with all negative null motion (left). Empty circles are child nodes under consideration for future searches.

three child nodes is opened, the child node corresponding to $u(t_k) = +1$ is selected for expansion. Since it is useful to identify a trajectory uniquely by the $\mathbf{u}_N$ vector required to reach its specific final state $\theta(t_n)$, I note that this first exploration is identified by $\mathbf{u}_N = [1\ 1\ \ldots\ 1]$. In all cases when a child node is selected for expansion the value of the cost function (5.7) for the remaining nodes is saved in a database of "open nodes" for use in potential future trajectory searches. In figure 5.4, the open circles represent open nodes which have not been investigated yet as part of a trajectory, but whose cost is known because of the exploration of a nearby trajectory. Filled circles represent nodes which are on a trajectory which has already been investigated and will not be considered in future trajectory searches.

The second trajectory begins again at the root node $\theta(t_0)$ and always selects the child node corresponding to $u(t_k) = 0$, no null motion. This trajectory is equivalent to the SDA pseudoinverse solution and is identified by the control vec-

tor $\mathbf{u}_n = [0\ 0\ \ldots\ 0]$.

The third trajectory begins at the root selecting the child $u(t_k) = -1$ at each decision node. This trajectory is identified by the control vector $\mathbf{u}_n = [-1\ -1\ \ldots\ -1]$.

By investigating these three simple trajectories, shown in figure 5.4, I quickly generate a large collection of nodes representing partial trajectories whose partial cost is known. The nodes whose children have not yet been expanded I call open nodes. The cost of the final node of each trajectory gives a measure of the relative worth of the trajectory. The best cost of all the completed trajectories is saved for comparison with future trajectories.

5.5.2 Directed Search Algorithm

Next, the directed search algorithm begins. At the start of each trajectory investigation, the algorithm selects the open node with the lowest cost from the database of potential nodes. For example, the first instance of directed search will choose from all of the open nodes from figure 5.4. This node can be thought of as having the highest "potential" for being on a low-cost trajectory, since the cost which has already been accrued on the trajectory up to that point is as low as possible.

Once the starting node is selected, a greedy search is performed where the child node with the lowest cost is selected for expansion, as shown in figure 5.5. Unselected sibling nodes are continually added to the database of future potential starting nodes. The open circles in figure 5.5 indicate nodes whose cost is known but are not yet

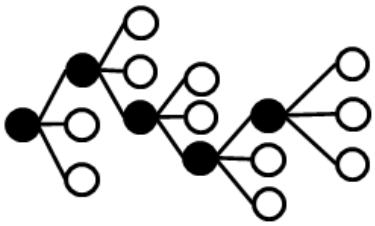


Figure 5.5: After expanding a parent node, the child node with the lowest cost defined by equation (5.7) is selected for expansion creating a depth-first search. The costs and locations of unselected siblings are saved for future consideration.

under investigation as part of a trajectory. The data from these nodes is saved for future use. As each node is expanded, its cost is compared to the cost of the current best trajectory. If the node currently under investigation already exceeds the cost of the current best, the algorithm restarts by selecting another starting node from the database. If the trajectory under investigation reaches the terminal state $\theta(t_n)$, its cost is compared to the cost of the current best trajectory. If the new trajectory is better than the current best, the new trajectory replaces the current best and its cost becomes the new comparison cost value. Then the process is repeated, starting by selecting a new starting node from the open node database as shown in figure 5.6. A flow chart of the algorithm is included in figure 5.7 for a visual explanation of the search process employed by the algorithm.

This search method is reminiscent of the optimal policy for finding a route through snow-bound Canadian roads in the Canadian traveler problem discussed in section 5.4.2. The first stage was to explore all zero-cost paths. Although the CMG prob-

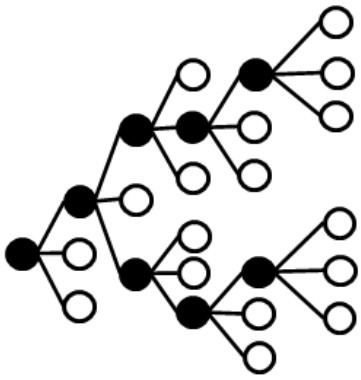


Figure 5.6: The open node with the lowest cost from figure 5.5 is selected as the starting node for the next depth-first search.

lem does not have zero-cost paths (indeed, the exploration or computation cost of propagating a node from $\theta(t_k)$ to $\theta(t_{k+1})$ is always the same), I begin the algorithm by performing some low-cost exploration by making $u(t_k)$ constant on the trajectory (figure 5.4). Then, as in the Canadian traveler problem, the next step is to select the open node with the highest potential to be on the best trajectory and explore it to the end (figure 5.5). Unlike the Canadian traveler problem, in the CMG problem I have the luxury of performing this last exploration several times since there is no cost associated with returning and restarting other than the cost of exploring the new trajectory (figure 5.6).

The algorithm in this thesis can be easily tailored to match the requirements of the real-time missions. For real-time application, the algorithm can continue improving the trajectory until the time is up and the satellite needs the algorithm to report a selected trajectory. I will use this idea with the "stage division" method employed

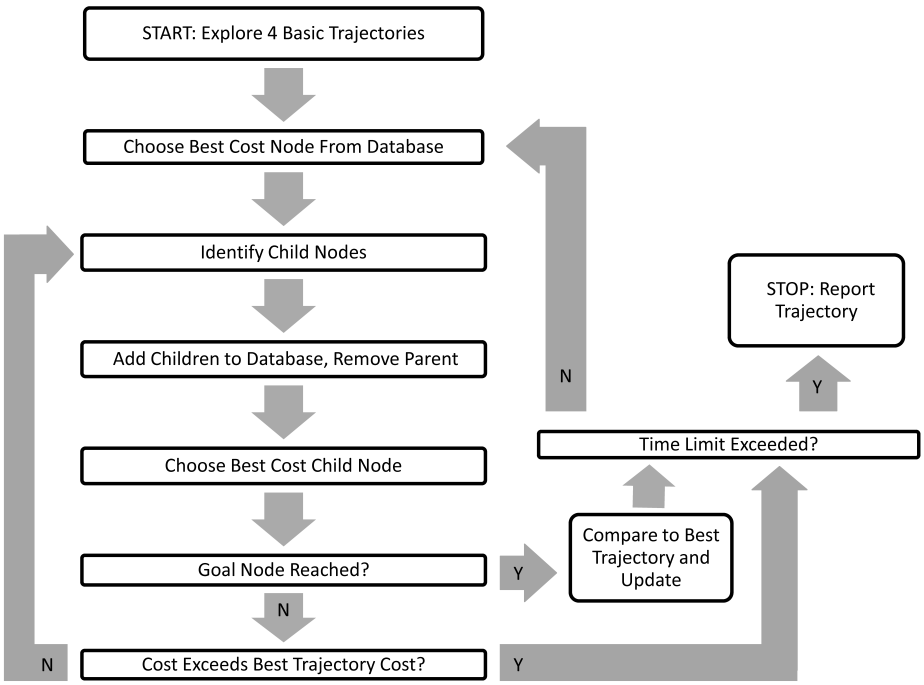


Figure 5.7: The search algorithm developed in this thesis.

by [12] to apply the search algorithm to real-time applications in section 6.

5.5.3 Offline Search Results

In this section, I present the results of using my search method in its basic form and compare it first to the results of computing the same trajectory using the Moore-Penrose (M-P) pseudoinverse of equation (3.14), and then to the results of using the Singularity-Robust (SR) pseudoinverse of equation (4.2). Finally, I compare my search algorithm with a global 2-point boundary value solver using DIDO.

First I compare the results of my search algorithm to the results when the satellite is controlled using the M-P Pseudoinverse to control the inverse kinematics in figure

5.8. This solution is computed using the M-P inverse in equation (5.3) with $u(t_m) = 0$ for all $t_m \in P_m$. I direct both methods to produce a trajectory which will accomplish a 30 degree pitch maneuver while following a designated sinusoidal torque profile $\dot{H}(t)_{desired}$. The solid lines show the trajectory selected by the search method of this thesis. In the top left plot, my search method achieves the 30 degree pitch maneuver, while the M-P inverse fails the mission. The plot in the top right explains why: this thesis' method selected a trajectory which succeeds in avoiding the singular state at $t = 10$ and so was able to follow $\dot{H}(t)_{desired}$ exactly. The M-P inverse produced a trajectory which fell immediately into singularity. Once in singularity, the M-P inverse commanded oscillating infinite gimbal rates (see bottom right) which the satellite could not physically accomplish and so instead it commanded $\dot{\theta}_{max}$. This means that $\dot{H}_{desired}$ and $H_{desired}$ were not followed (see center left and center right).

Secondly I compare the results of the same trajectory with the effect of using the Singularity Robust (SR) inverse of [1] in figure 5.9. This solution uses the SR inverse to solve equation (5.3) with $u(t_m) = 0$ for all $t_m \in P_m$. The SR inverse is incapable of escaping the singularity (top right) but effectively skirts it by adding torque error (center left and center right). By preventing the system from actually entering the singular state, the SR inverse avoids commanding gimbal rates which exceed $\dot{\theta}_{max}$ (bottom right), but still does not achieve the desired 30 degree pitch rotation.

This thesis' method produced a significantly improved trajectory compared to the results of using either the M-P inverse or the SR inverse.

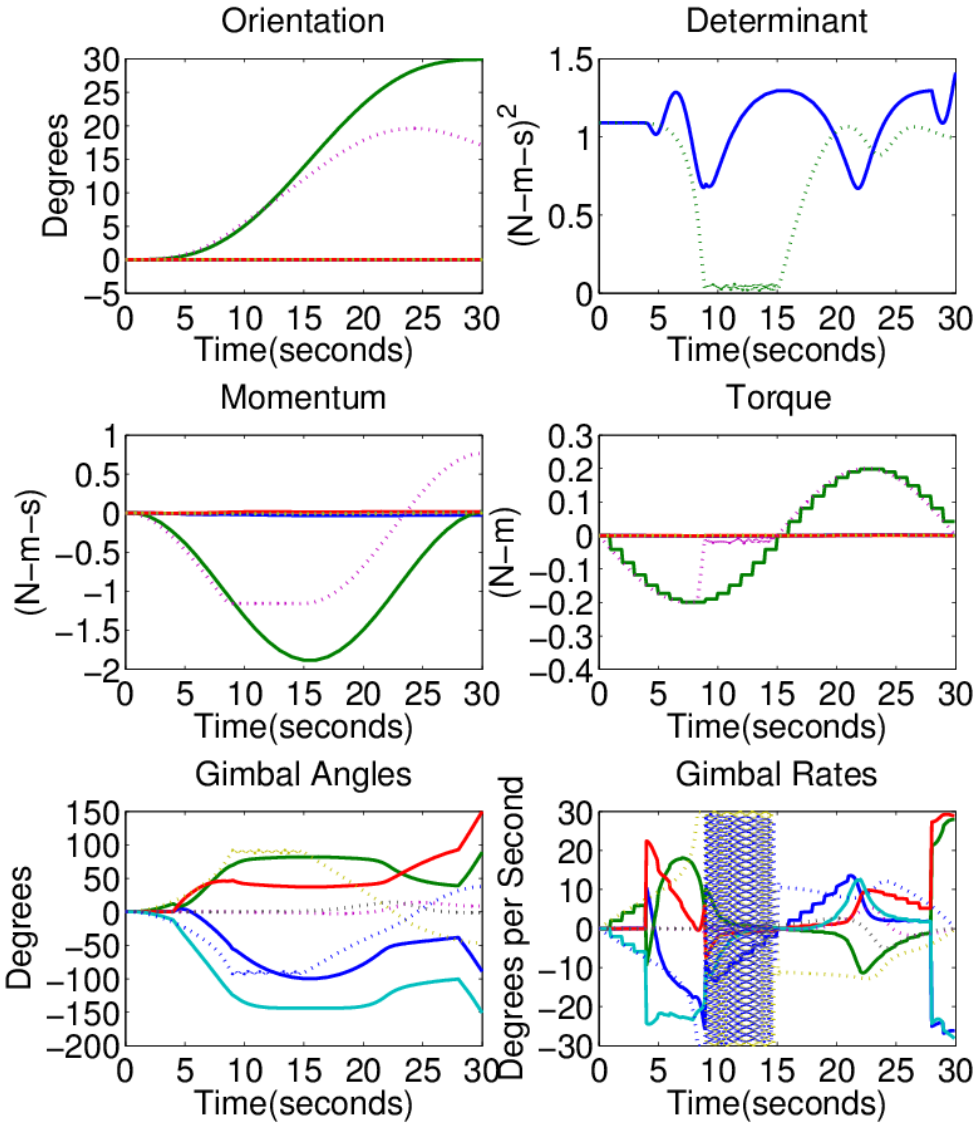


Figure 5.8: A 30-degree pitch maneuver. The dotted line is the Moore-Penrose (M-P) Pseudoinverse, and the solid line is the Search Method of this thesis. Near a singular state, the M-P inverse causes wild oscillations which mean that $H_{desired}$ and $\dot{H}_{desired}$ are not followed and the final orientation is not attained. However, the search method of this thesis completes the maneuver successfully.

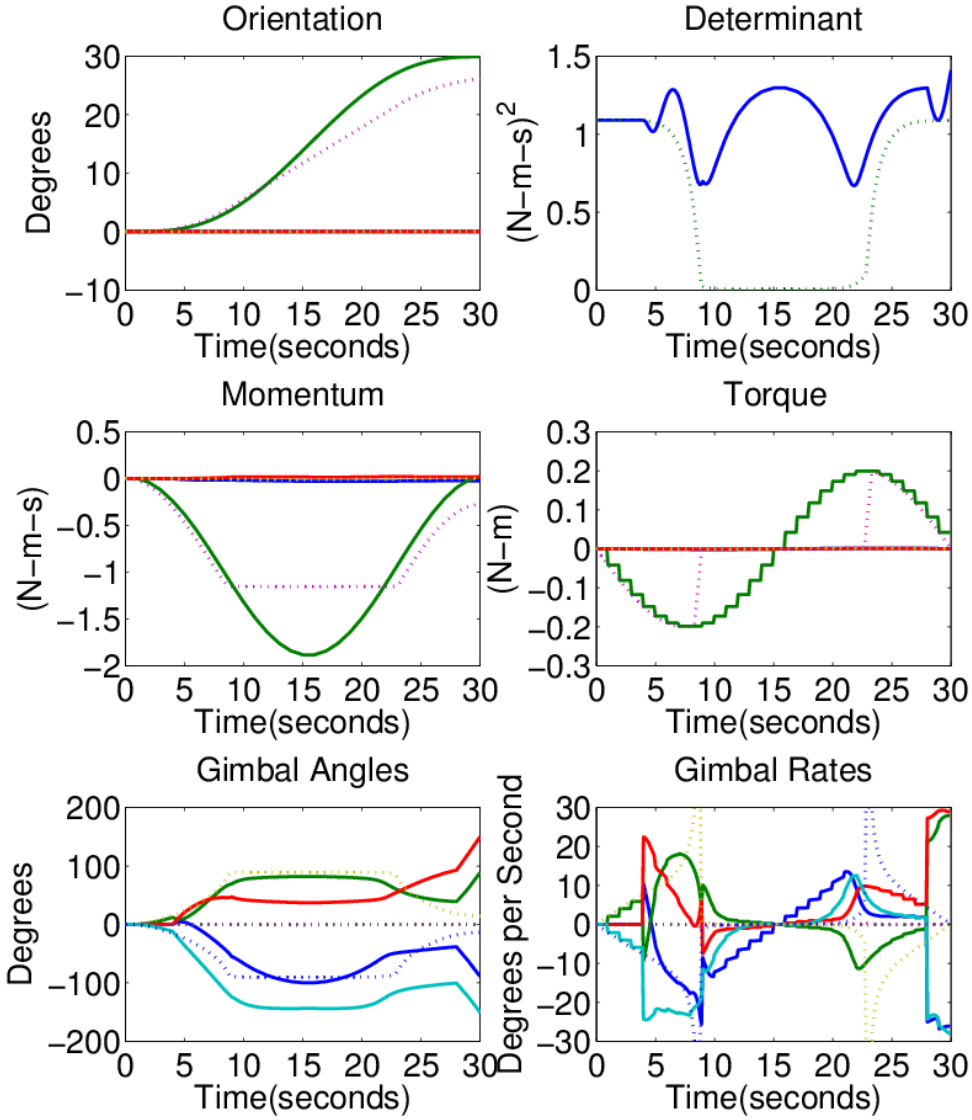


Figure 5.9: A 30-degree pitch maneuver. The dotted line is the SR inverse, and the solid line is the Search Method of this thesis. Near singularity, the SR inverse adds torque error to keep the gimbal rates within limits; so $H_{desired}$ and $\dot{H}_{desired}$ are not followed and the final orientation is not attained. My search method avoids singularity and accomplishes the desired mission profile.

Next, I compare the results of my search method with the results of using DIDO [7, 24] to solve the complete 2-point boundary value problem of equation (3.11) with no path constraints on the state variables. Note that the problem (3.11) which is solved using DIDO and problem (3.20) solved by this thesis' search algorithm are different. I chose to apply DIDO to (3.11) to be able to achieve higher determinant values by dropping the path constraints.

Since my search method is limited by the amount of cpu time allotted for the search, whereas DIDO cannot be guaranteed to produce a result in a specified amount of time, it is difficult to compare them directly. However, unlike my algorithm, the DIDO implementation can be set to search for a trajectory which achieves a certain criteria for the determinant of equation (3.9). Therefore, to compare the results of the application, the DIDO implementation was timed to see how long it takes it to find a feasible trajectory satisfying various minimum bounds on the determinant. Specifically, DIDO was used to solve (3.11) with the added constraint

$$m(\theta(t)) \geq m_{\min} \quad t \in [t_0, t_f],$$

where $m_{\min}$ is set to one of the values in $\{0.5, \ldots, 1.0\}$. In all cases, DIDO was run using 40 collocation points. DIDO uses the Legendre-Gauss-Lobatto nodes. The results are shown in table 5.2.

Then, the search algorithm of this thesis was implemented using various time constraints, but without the lower bound on the determinant, which cannot be applied using my solution method. My search algorithm was applied with 40 time steps for

Determinant bound	DIDO time (s)
0.5	97
0.6	65
0.7	54
0.8	74
0.9	70
1.0	125

Table 5.2: Time it took DIDO to solve the optimal control problem (3.11) with an added lower bound on the determinant.

each optimization. The results are shown in table 5.3.

By comparing the results of the two solution methods, it is possible to see that whereas the DIDO implementation was capable of finding trajectories which satisfied high performance requirements on the determinant, it did so at the cost of computation time. One of the major drawbacks of using a solution method such as DIDO is the sporadic nature of the time required to compute a solution, as shown in table 5.2, where the time required does not increase linearly with the constraint difficulty.

The search method of this thesis, on the other hand, was able to deliver similar performance results on the determinant after only 20 seconds of search as DIDO produced in 97 seconds of search. Additionally, as the allotted search time increases, the performance results delivered by the algorithm increase in a predictable way until

Search time (s)	Minimum determinant
20	0.496
30	0.592
40	0.714
50	0.714
60	0.714

Table 5.3: Minimum determinants found by the search algorithm of this thesis, for various maximum search times.

the search method reaches its "peak" performance for this test case by delivering a trajectory with minimum determinant value of 0.714 after 40 seconds of searching. Because the discrete problem formulation limits the number of trajectories which are under consideration, the search algorithm cannot improve the trajectory even when more search time is allotted. Therefore, for higher search times, DIDO is capable of producing a result with a higher minimum determinant value on the trajectory.

The graphical results of the DIDO search with lower determinant bound of 0.6 and the results of the thesis search method with time limit of 30 seconds are shown in figure 5.10. Coincidently, the two methods took similar amounts of time to produce trajectories with similar performance as measured by determinant trajectory. However, it is easy to see that the DIDO trajectory followed a different attitude profile than the one required by the mission assumption of this thesis.

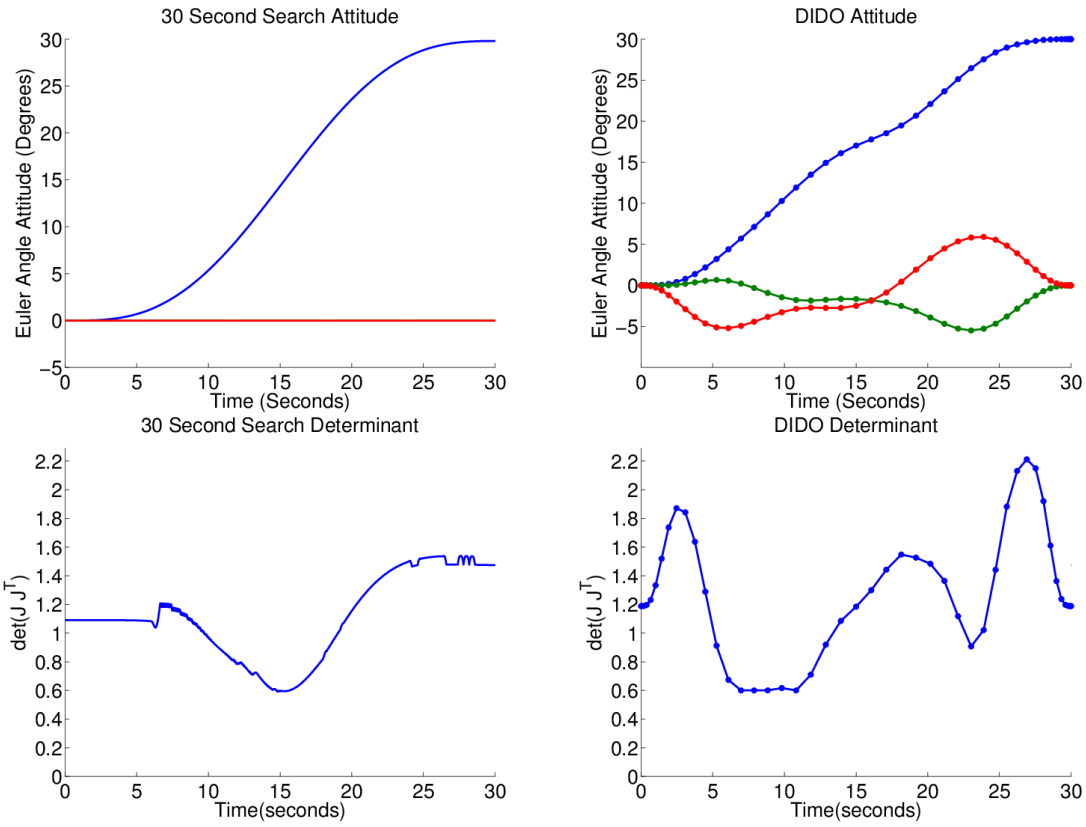


Figure 5.10: Comparison of DIDO implementation and Thesis search method. Top Row: Attitudes for each maneuver. The attitude profile of the DIDO implementation is unconstrained. Bottom Row: The two methods achieve minimum determinant values of 0.6.

Chapter 6

Real Time Search Feasibility

Having developed an improved discrete search-based optimization method to find singularity-avoiding trajectories for satellite rotation, it is now possible to investigate the critical question of real-time application.

To that end, I developed the real-time framework necessary to apply my search algorithm for two separate theoretical real-time mission requirements. In section 6.1 I explain the first which I shall call point-and-click and which is suitable for rapid reorientations for a series of maneuvers. In section 6.2 I explain the second real-time framework, based on the "stage division" method pioneered by Ikaida et al. [12] which is suitable for real-time trajectory optimization for long trajectories when multiple re-optimizations are possible during the maneuver.

In section 6.3, to demonstrate the feasibility of my method for real-time application, I provide the results of several Monte Carlo simulations testing the algorithm

under various conditions. When the conditions are within the expected performance range of the algorithm, my method succeeds in providing trajectory improvement in a simulated real-time environment. Finally in section 6.4, I present the results for a single test case to demonstrate the success of the algorithm in finding an improved trajectory for the difficult roll maneuver.

6.1 Point and Click

The first real-time application that I applied the new algorithm to I name Point and Click because of the method would allow a satellite to adjust its attitude to take aim for a photograph. An earth imaging satellite might have this type of mission if it were required to take a series of images of different locations on the earth back-to-back with a hard-mounted camera. So the 'aiming' of the camera for multiple shots is the mission goal of the following real-time application.

Before the satellite begins to move to take the first shot, I assume it has a limited amount of time to select a rotation trajectory to reorient and aim the camera. For this test, I use $h_{CMG} = 1$ N-m-s and rotation rate 1 degree per second.

I gave the satellite $t_{f1} - \epsilon$ seconds to find the trajectory from the final state at t_{f1} to the state required for the second photo at t_{f2}. If the satellite uses less than the allotted time, it can use the extra time in the third search. When the satellite reaches $t_{f1} - \epsilon$, the algorithm reports the best trajectory it has found for the next stage. When the satellite reaches t_{f1}, it takes the first photo, and the satellite can

immediately begin following the selected trajectory towards the second photo state at t_{f2}. Then the algorithm has $(t_{f2} - t_{f1}) - \epsilon$ seconds to select the trajectory which will carry the satellite to its third photo state, and so on.

The results of this thesis' search algorithm on a triple point-and-click maneuver is shown in figure 6.1 below. The three orientations which the satellite will achieve from the starting attitude of $e_0 = [0\ 0\ 0]^T$ are $e_{f1} = [0\ 30\ 0]^T$, $e_{f2} = [0\ 0\ 0]^T$ and $e_{f3} = [0\ -20\ 5]^T$. For each trajectory, the time required for the satellite to execute it dictates the amount of time which the satellite has to search for and choose the next trajectory. In the case given in 6.1, the first maneuver is 30s, so the satellite has 25s to select and choose the trajectory it will follow from t_0 to $t_{f1} = 30$. Here, $\epsilon = 5$s was ample time for the code to report its selected trajectory and restart the search for the next trajectory before 30s had elapsed. In this thesis' trial, the algorithm used all 25s to continue improving the first trajectory. After 30s, the satellite implements the chosen trajectory and has 25s to select the second, 30s trajectory. If the satellite does not use all 25s, it can 'save up' time and to use when selecting a challenging trajectory. For example, in this thesis' implementation, the second and third trajectories were easily selected after only 2 and 3s, respectively. In those trajectories, the determinant stayed high and so there was no reason to continue the search. In the first trajectory, the pseudoinverse solution alone dives towards a hyperbolic singularity and so the algorithm required some effort to avoid it. As can be seen in figure 6.1, my method succeeds in maintaining the system away from singular configuration.

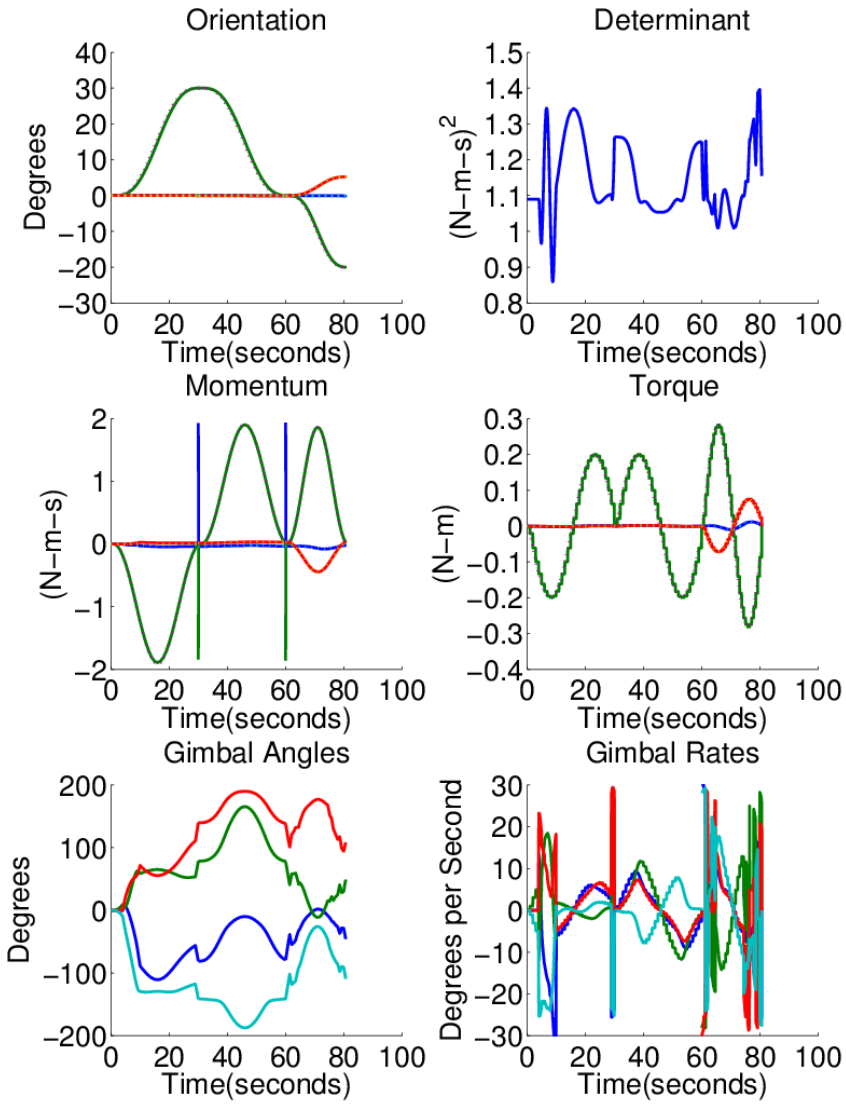


Figure 6.1: A real-time calculation of a triple maneuver. The satellite has 30s to choose the first 30s trajectory, then 30s to choose the second trajectory, and 20s to choose the third. The satellite successfully maintains its distance from singularity.

6.2 Stage Division

Based on the work of Ikaida et al. [12] for noise-abatement real-time trajectory optimization for helicopters, this thesis develops the following stage division framework to apply the discrete search algorithm to optimize long trajectories efficiently in real time. Stage division is similar to model predictive control or receding horizon control (see [18]), except that for stage division the "horizon" is fixed at t_f and instead the time discretization scheme scrolls or recedes while the search algorithm performs a sequence of discrete optimizations.

Ikaida et al. noted that the calculation required to optimize a densely discretized lengthy trajectory for the helicopter was far too slow for real-time implementation. For that reason, they developed a method they call "stage division" to more intelligently distribute the optimization nodes to allow reduced computation time for real-time application [12].

The explanation is as follows. The first optimization calculation is begun before the satellite reaches the starting state at t_0. It uses a dense discretization for the decision nodes on $t_k \in [t_0\ t_{crit,1}]$ of the trajectory, and a rough discretization of the remainder of the trajectory until t_f. When the satellite reaches $t_0 - \epsilon$, the best trajectory is reported and at t_0 the satellite immediately begins following this trajectory. The algorithm has $t_{crit,1} - \epsilon$ seconds to compute the next portion of the trajectory which is densely discretized from $t_{crit,1}$ to $t_{crit,2}$ and roughly discretized until t_f as before. At $t_{crit,1} - \epsilon$ the best trajectory is reported and immediately implemented

when the satellite reaches $t_{crit,1}$. The process continues until the helicopter reaches t_f. In this way, the trajectory gradually becomes more optimal in spite of the limited nature of the search. The node density and spacing of the values of t_{crit} can be tailored to the mission requirements and capabilities of the satellite. The optimization also continually accounts for changes in the state of the system (disturbances) and optimizes the trajectory gradually in real-time by prioritizing the nodes closer to the current state of the satellite while still considering the nodes which are farther away.

This feature is an advantage over the point-and-click method which does not seek an optimal end state at the end of each sub-trajectory. So the algorithm may select a trajectory for the first photo which leaves the satellite in a challenging state for the start of the trajectory for the second photo. However, for the point-and-click method it is assumed that the attitude required for subsequent photos is not known in advance.

The stage division method developed by Ikaida et al. was successfully employed in real-time for helicopters by using a simplified computational model, this thesis shows that the search-based method is extremely well suited for application to the stage division method with little modification. Since the search-based method can continue exploring trajectories until the satellite reaches t_{crit}, the problem of balancing computation time with the goodness of the selected trajectory has a natural solution in the stage division method. The algorithm continues searching new trajectories and comparing them to the current best until the satellite approaches the critical time

and demands its trajectory update.

The results of the Stage Division method when applied to a trajectory search in a CMG problem are shown in figure 6.2. Using a 30s rotation from $e_0 = [0\ 0\ 0]^T$ to $e_f = [0\ 30\ 0]^T$, the problem is divided into five stages with $t_{crit} = [6,\ 12,\ 18,\ 24,\ 30]$. While the satellite is employing the densely discretized portion of the selected trajectory, the algorithm is searching to improve the selected trajectory for the next stage.

At the start of the process, the algorithm has 6 seconds to select the trajectory it will follow from $t_0 = 0$ to $t_1 = 6$, and 1s is removed for processing. The algorithm works during these 5 seconds to select a complete 30s trajectory which will actually only be followed by the satellite for the first 6s. After 5s, the algorithm reports the best trajectory which it has discovered and it is immediately applied at $t_0 = 0$. Then the algorithm has 5s to select the second trajectory from $t_1 = 6$ to $t_3 = 12$, and so on. In searching for the trajectory shown in 6.2, the algorithm continued searching for trajectories for the full 6s during the search for the trajectories for the first three stages, but for the final two stages the algorithm requires only 1.5s and 1s to find the discrete optimum. The complete resulting trajectory from this search appears in 6.2, but the evolution of the determinant of the various trajectory searches is of particular interest and so appears in figure 6.3.

In the first plot in figure 6.3 below, showing the determinant trajectory from the first stage search, the satellite has a very limited time to address a poor CMG gain situation as the pseudoinverse solution dives into a hyperbolic singularity and so the

selected trajectory is less than ideal. The trajectory avoids losing all CMG gain but has a poor outlook for the rest of the trajectory. In subsequent stages, the singularity at $t = 26$ is addressed through null motion and the actual CMG gain of the satellite at time $t = 26$ is nearly ideal with a value close to 1. For comparison, the sixth panel of figure 6.3 contains a comparison between the result of this thesis' search method and the SDA singularity avoidance method of Ford and Hall [8] discussed in section 4.2.

This real time application is suitable for short trajectories such as the 30s trajectory shown in figure 6.2 as well as for longer trajectories closer to the intent of Ikaida et al. when they developed the stage division method for lengthy helicopter landing trajectories. It allows a much more intensive optimization than a straightforward application of the discrete search method. Recall the number of decision nodes N that can be considered by the search method is limited by the computational capabilities of the platform. Using a tertiary search tree, I have 3^N trajectories to consider. In general, it was found in this thesis that between 30-40 nodes provided a balance between computational time and optimality of the result. However, by employing the stage division, the distribution of nodes means that the number of nodes used to find the actual final trajectory is much higher!

In the examples provided in figures 6.2 and 6.3, the dense discretization has 30 nodes while the sparse discretization has only 10. So each discrete search is limited to considering 3^{40} trajectories. However, the algorithm is set up so that a satellite

using it would only traverse the part of the trajectories which are densely discretized. In this case, the final trajectory was optimized over 30 nodes in the dense portions of each of the 5 stages for a total of 150 nodes. It would be computationally impossible to consider 3^{150} trajectories, but the algorithm was able to obtain some of the freedom and accuracy of considering that quantity of trajectories by dividing the work into stages.

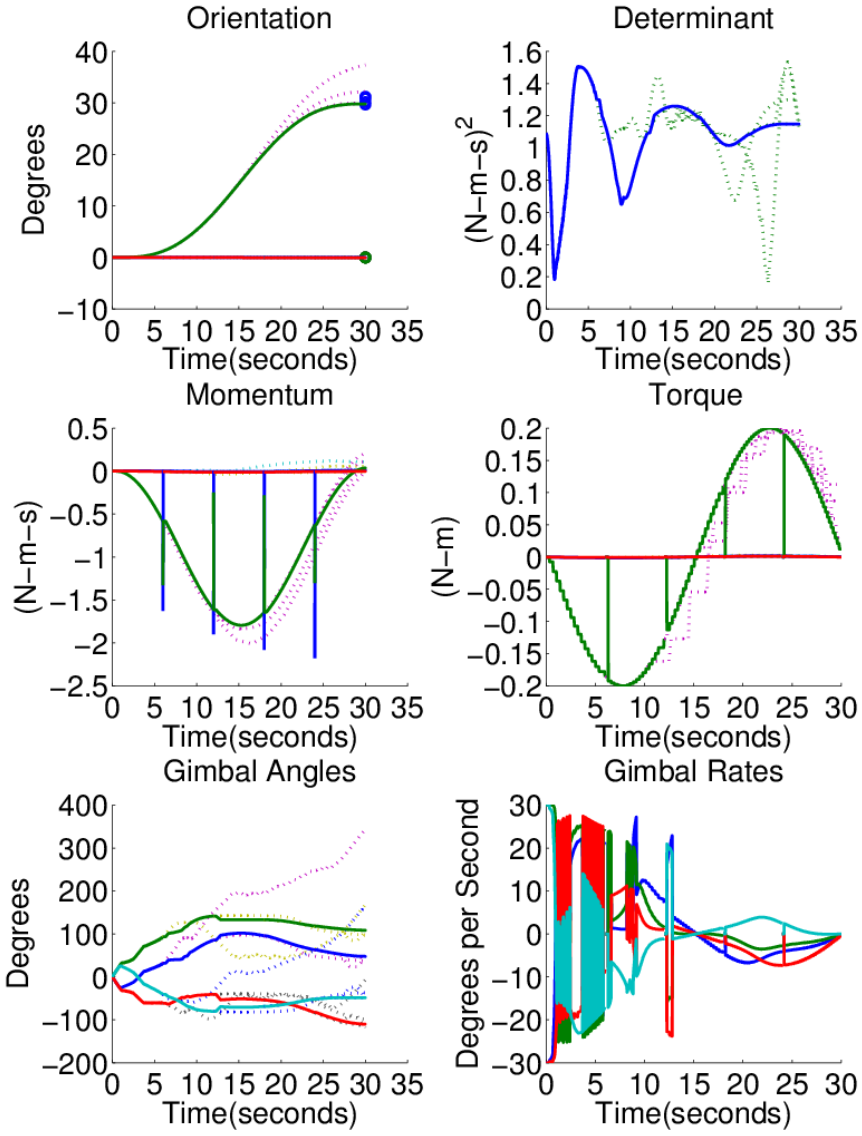


Figure 6.2: A real-time calculation of a stage division calculation for 30 degree pitch rotation. The algorithm performs the optimization 5 times over the trajectory to continually improve it in real time. The subsequent trajectories are a significant improvement over the algorithm's first choice.

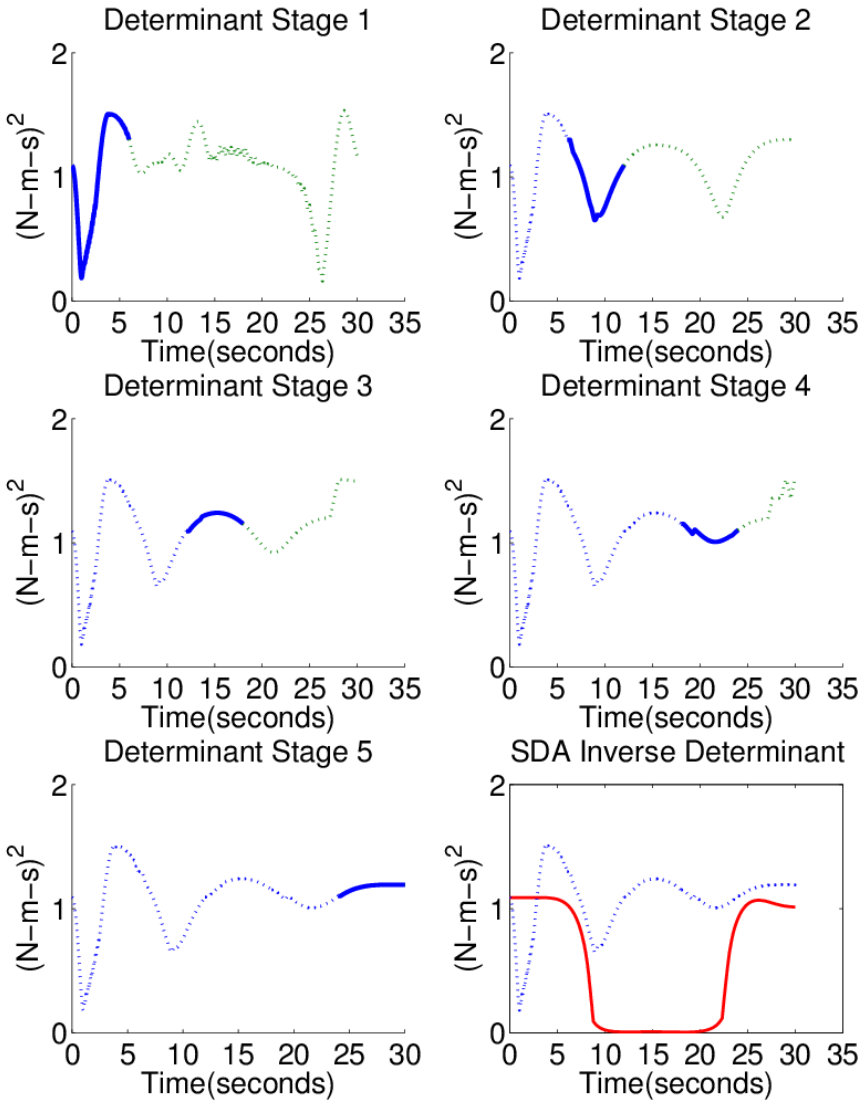


Figure 6.3: The determinants of the trajectories selected during each stage for a 30 degree pitch rotation. The solid line is the portion the satellite follows during that stage. Dotted lines are past or predicted trajectories. It is harder avoid singularities in early stages than in later stages.

6.3 Monte Carlo Simulation Results

In order to demonstrate the feasibility of this thesis' proposed search algorithm both in finding improved trajectories and in improving a selected trajectory in real-time, I will next present a series of Monte Carlo simulations, designed to demonstrate the robustness of the algorithm for large samples of randomized initial and final conditions.

The first four Monte Carlo simulations are performed using a desktop computer running MATLAB with AMD Athlon^{TM}64×2 Dual Core Processor 6000+, 3.01 GHz, and 2.00 GB of RAM.

First are two Monte Carlo simulations, designed to show the difference in performance by the algorithm when the mission requires only slow rotation rate compared with when the mission requires more challenging high rates of rotation. In each simulation, I apply the stage division method using five divisions. However, the computation time to select each stage of the trajectory is the same for each trajectory, and is not meant to simulate real-time stage division, where the computation time to select the next stage would be required to be less than the actual flight time of the current stage. Instead, in order to easily compare a large sample of simulations, the initial computation time for the first stage is uniformly set at 40 seconds, with 10 seconds allowed for each of the subsequent four stages.

The average rate of each trajectory in figure 6.4 is 5 degrees per second, and in figure 6.5 the average rate is 10 degrees per second. Because it is more difficult

for the algorithm to avoid singularities when the satellite approaches them faster, I adjusted t_f for each trajectory based on for initial eigenangle defined by equation (A.4) by setting $t_f = \frac{eigenangle}{rate}$. That way, the trajectories compared in each Monte Carlo simulation are equally difficult for the algorithm to improve. Initial and final attitudes are defined using a randomized Euler angle $\phi \in [-\pi\ \pi]$ and the Euler axis defined as a random unit vector to define the Euler parameter using equation (3.1). For comparison purposes, in the following graphs the time has been normalized so that the stage divisions and t_f for each trajectory are in line with each other. In addition, each of the five stages is slightly off-set so that it is possible to separate them visually.

In figure 6.4 are the results of a large sample of trajectories with slow rotation rates. With an average maneuver rate of 5 degrees per second, the algorithm has adequate capability to detect singularities in advance and avoid them using null motion. With a high gimbal rate limit of 60 degrees per second, even trajectories with low determinant have high tracking accuracy, and all 745 simulations are successful in completing the tracking mission. I note for later comparison that 11 of 745 or 1.5% have minimum determinants less than 0.5. However, the objective function had a secondary objective to minimize the time spent in the vicinity of a singular state, and the success of this is evidenced by the sharpness of the few determinant trajectories which approach near-singular states.

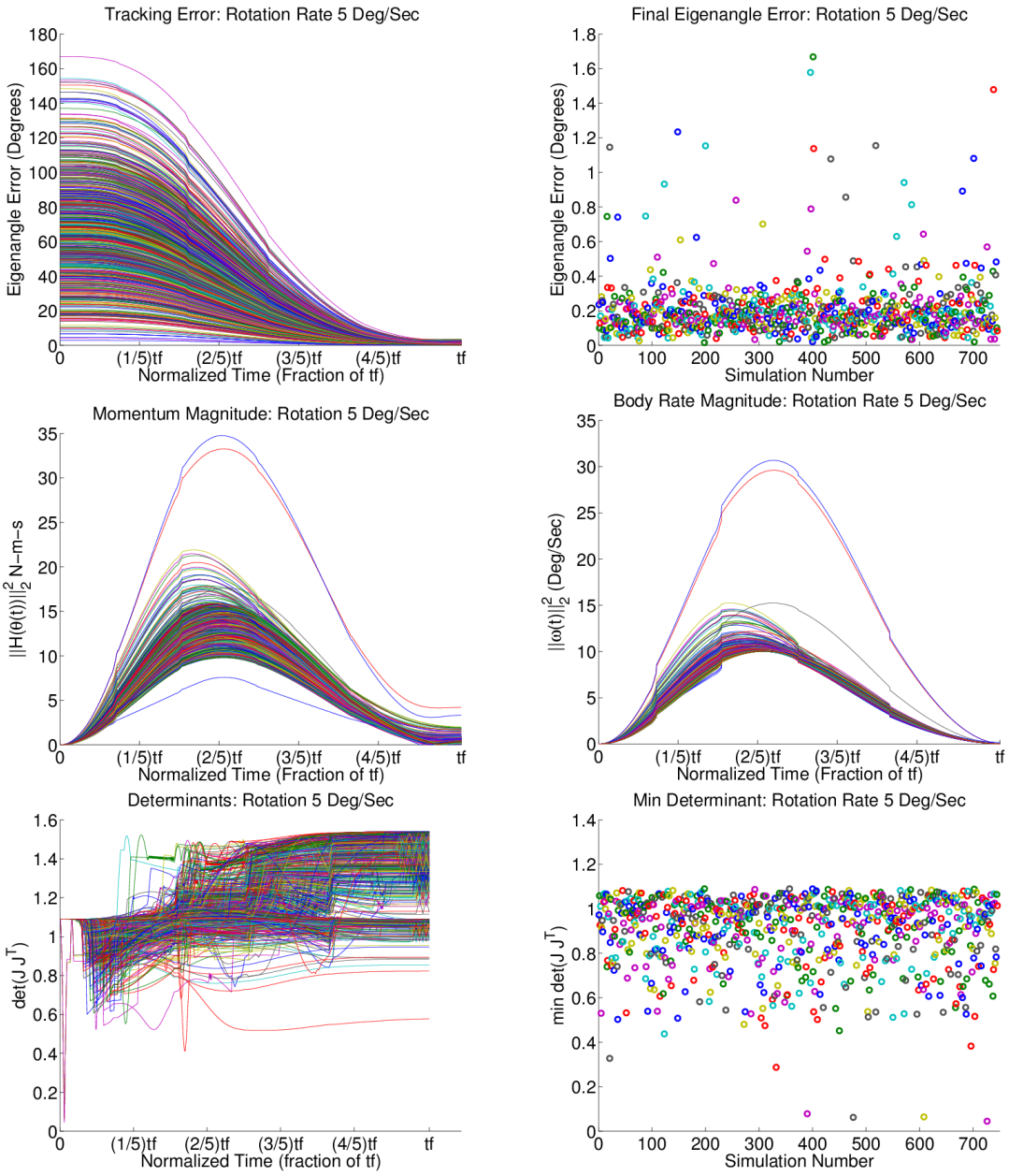


Figure 6.4: Results of 750 Monte Carlo simulations with randomized initial and final attitudes, using stage division with 5 stages, rotation rate 5 deg/sec and $h_{CMG} = 15$ N-m-s, and gimbal rate limit 60 degrees per second. The discrete search based method finds excellent trajectories which minimize the time spent in low determinant states.

In figure 6.5, I present the results of the same 750 trajectories with more challenging conditions. Here, minimum rotation rate in each axis is 10 degrees per second, which is the limit of where I expect the algorithm to perform well given the capabilities of the satellite it is being tested on. The satellite has CMGs capable of providing $h_{CMG} = 15$ N-m-s momentum each. In general, it is more difficult for the algorithm to detect and avoid singularities using null motion when the satellite approaches them more rapidly since proportionally less of $\dot{\theta}_{max}$ is available to be applied as $\dot{\theta}_{null}$. The effect is evident as 40 of 750 simulations, or 5%, have minimum determinants less than 0.5. However, the algorithm succeeds in limiting these instances of low determinant to a sharp valley to limit the amount of time spent in a near-singular state. Only a few outliers have higher final tracking error, with a maximum of 5.5 degrees, compared with mean 0.55 degrees. The final tracking error will be dealt with the control mechanism of the satellite, which is designed to control small attitude adjustments such as these outliers.

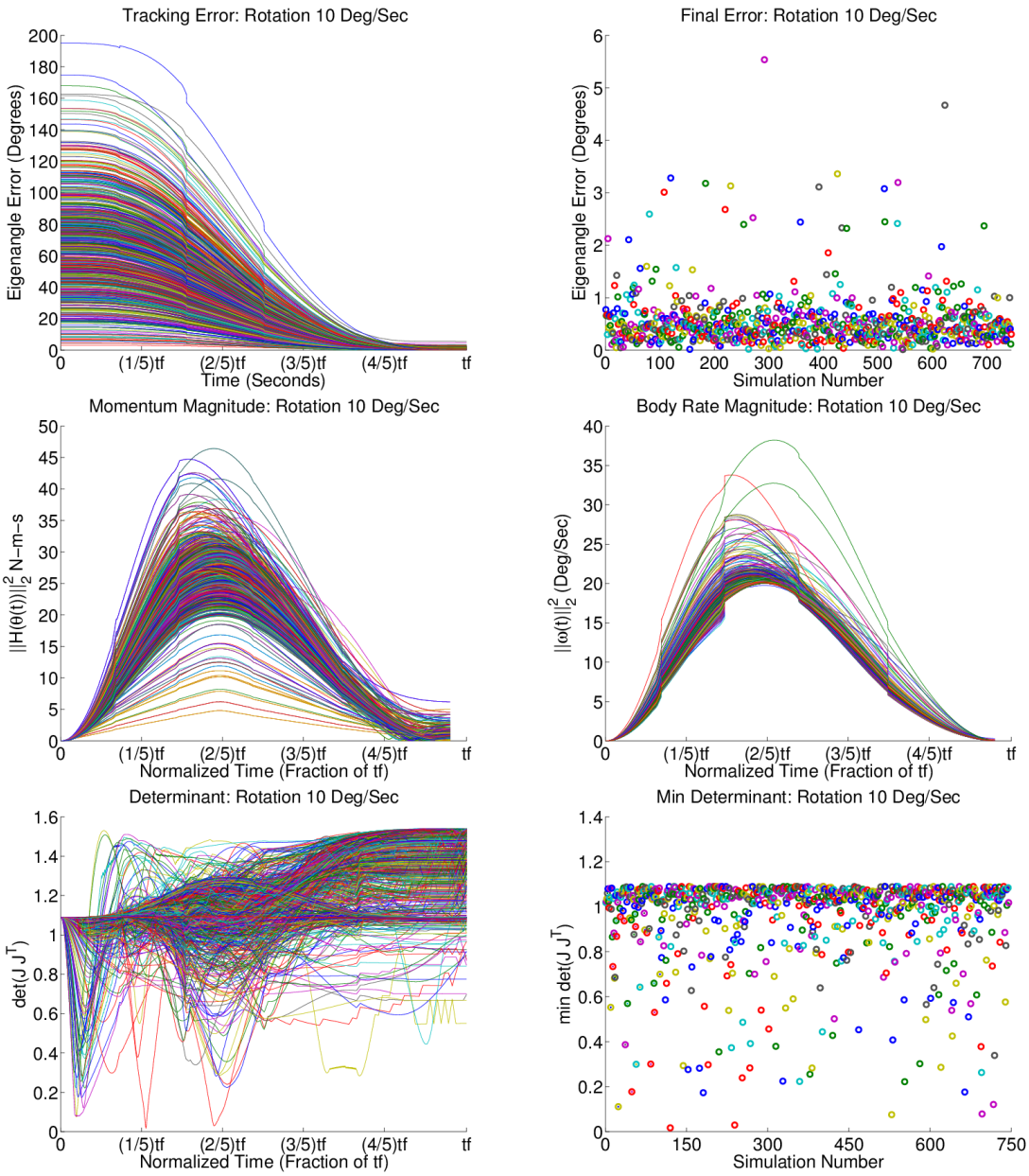


Figure 6.5: Results of 750 Monte Carlo simulations with randomized initial and final attitudes, using stage division with 5 stages, rotation rate 10 deg/sec and $h_{CMG} = 15$ N-m-s, and gimbal rate limit 60 degrees per second. Low determinant approaches are sharp when the search method succeeds in reducing time spent in low-gain states.

When a similar Monte Carlo is performed using the most challenging conditions with $h_{CMG} = 15$ N-m-s CMGs at an average body rate of 10 degrees per second in each axis, but with a challengingly low maximum gimbal rate constraint of 30 degrees per second, the undesirable effect of low CMG gain trajectories is clearly seen, as shown in figure 6.6. While the median final eigenangle error for these 183 trials is 0.53 degrees, and the mean is 0.85 degrees, the trajectories which encountered low CMG states have higher tracking error because of the lowered gimbal rate maximum. At low CMG gain, the desired gimbal rates to follow $\dot{H}(\theta(t))$ exceed the limit; so the actual applied gimbal rates cannot achieve $\dot{H}(\theta(t))$ and torque error is incurred. The final eigenangle error in these cases is much higher. Although in practice a satellite's capabilities would be better tailored to its mission so as to prevent this occurrence, these examples are helpful to demonstrate what happens when the algorithm is set to work on difficult cases with inadequate satellite capability. Here, 22 trajectories of 745 or 2.9% encountered CMG gain below 0.5. This demonstrates the tradeoff between low gimbal-rate limit and tracking accuracy, and also shows the need for intelligent choice of null motion since applying null motion which produces the maximum possible $\dot{\theta}$ may not result in optimal trajectories.

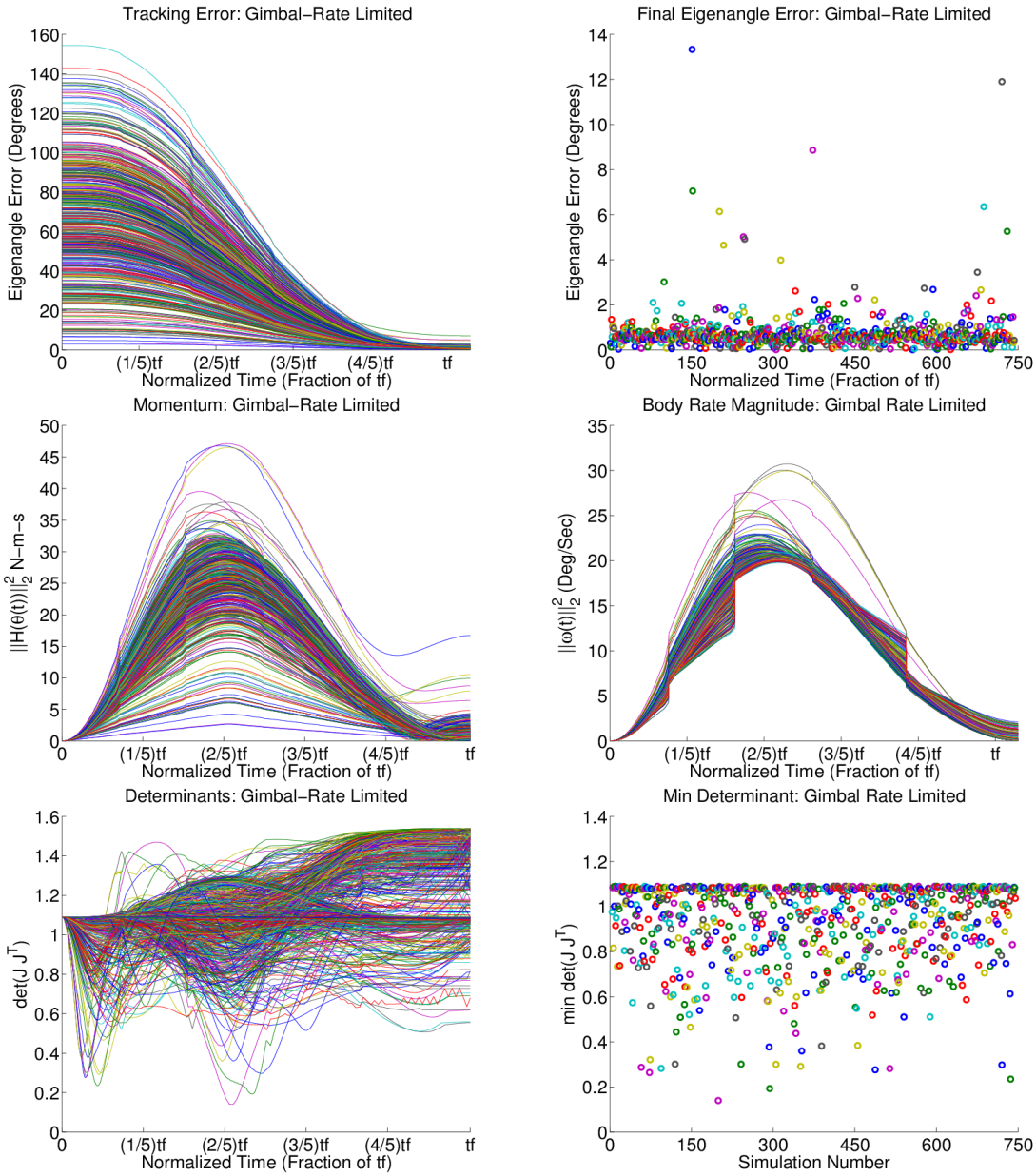


Figure 6.6: Results of 750 Monte Carlo simulations with randomized initial and final attitudes, using stage division with 5 stages, rotation rate 10 deg/sec and $h_{CMG} = 15$ N-m-s, and gimbal rate limit 30 degrees per second. Most trajectories have reasonable determinants, but the tracking accuracy suffers because of the low gimbal rate limit.

Similar results are seen when the algorithm is stressed beyond its practical limits in a different direction as shown in figure 6.7. I use the same size CMGs with $h_{CMG} = 15$, and the standard gimbal rate limit of 60 degrees per second, but require the maneuvers to be accomplished at an average body rate of 15 degrees per second. This performance is beyond what a satellite with total CMG momentum of 60 N-m-s should be expected to handle, and the results are less than desirable. As expected because of the fast rate required, the minimum determinants of the trajectories are very low with an average of only 0.62 and 51 of 250 trajectories or 20.4% have minimum determinant less than 0.5. Also, unlike previous cases where the algorithm allowed a low-determinant trajectory but minimized the time spent in low determinant states, here the algorithm allows 'rounded' valleys which indicate that the trajectory is hovering in near-singular or singular states before escaping. This failure basically comes down to the capabilities of the satellite, although these Monte Carlo trajectories are time-limited with the first stage allowing 40 seconds of computation time and subsequent stages allowing 10 seconds of computation time. For challenging maneuvers, longer computation times will allow the algorithm to find and select better trajectories. In spite of low determinants, the high gimbal rate limit of 60 degrees per second means that the final errors only reach as high as 7.5 degrees, and the mean final error in this case has increased to 0.90 degrees, which is greater than the mean error in the gimbal rate limited case of figure 6.6. To effectively use this algorithm in a satellite, then, it is necessary to carefully match the system requirements and capabilities.

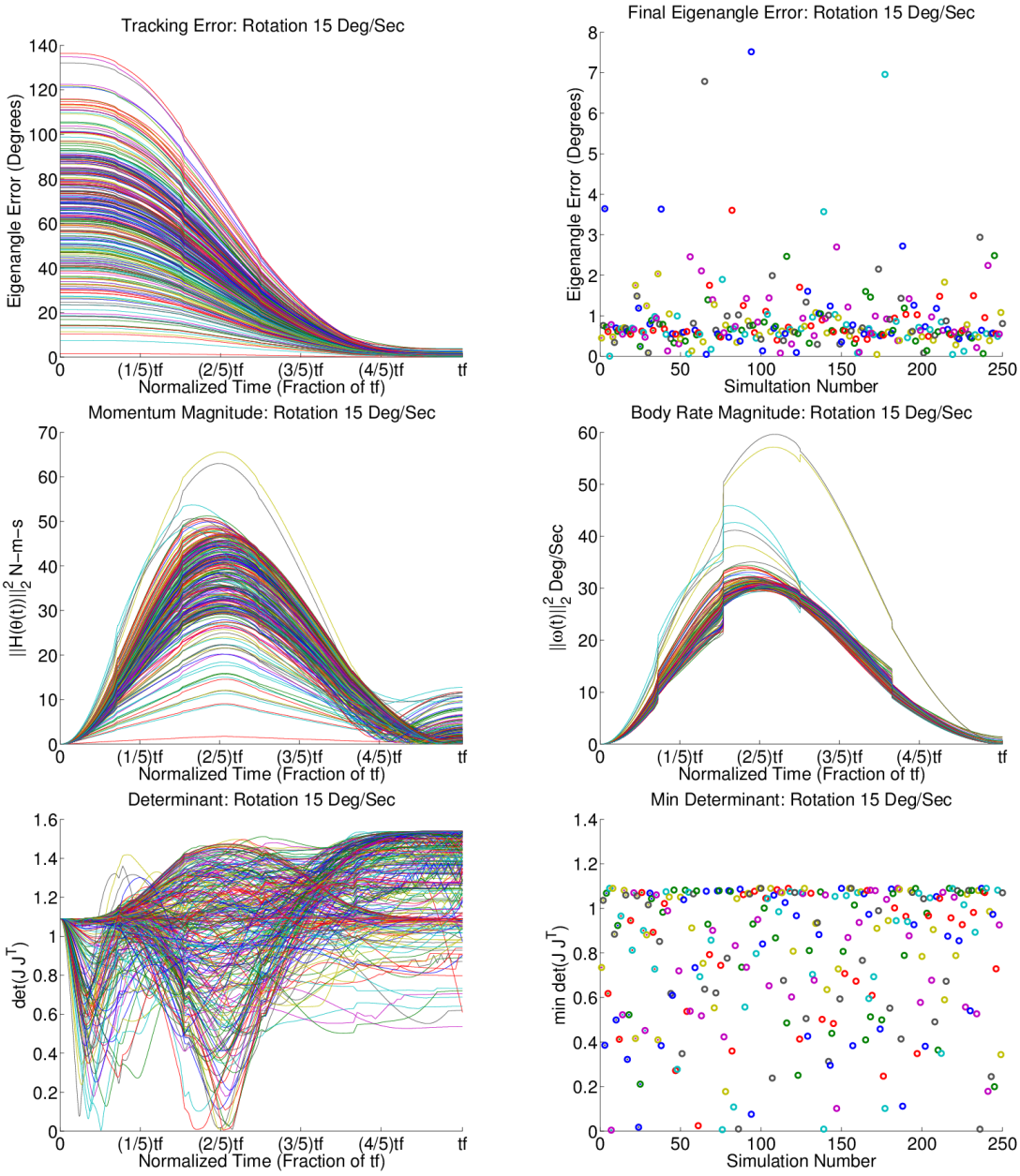


Figure 6.7: Results of 250 Monte Carlo simulations with randomized initial and final attitudes, using stage division with 5 stages, high rotation rate 15 deg/sec and $h_{CMG} = 15$ N-m-s, and gimbal rate limit 60 degrees per second. As expected, several trajectories hover in low determinant states and contribute to high tracking error.

Finally, I present the results of a Monte Carlo battery using simulated real-time stage division method in figures 6.8 and 6.9. These simulations were performed using a desktop computer running MATLAB with Sun Microsystems Ultra20 M2 Dual-Core AMD Opteron Processor 1214 and 3 GB memory. As these two real-time simulations were performed on a different computer than the previous, these are not meant to be compared directly to the preceding Monte Carlo simulations. Since the search method is limited by computation time, the computational capabilities of the platform affect the results. On a slower computer, less optimal trajectories will be found because less of the search space can be investigated in the same time. A faster computer will generate better trajectories for the same conditions. This is one of the greatest strengths of the algorithm for application to real-time satellite guidance, as the method can be tailored for use by satellites with either limited or more robust computational capabilities.

In the following simulations, I allow an initial computation time of 40 seconds for the first stage, as in the preceding Monte Carlo simulations. However, for all subsequent stages, the computation time to select the trajectory for the next stage is limited to less than the time length of the current stage. The average maneuver rate is 10 degrees per second. In the first Monte Carlo shown in figure 6.8, the gimbal rate limit is 60 degrees per second. In the second Monte Carlo of figure 6.9, the gimbal rate limit is 30 degrees per second. For both Monte Carlo simulations, the trajectories are chosen with randomized Euler parameters q_0 and q_f defined by randomized Euler

angle $\phi_0 \in [-\pi/4 \ -\pi/8]$ and $\phi_f \in [\pi/8 \ \pi/4]$ (see equation (3.1)), so as to limit the smallest initial eigenangle error which would be tested in the Monte Carlo simulation. This is because for a very fast maneuver it would not make sense to use stage division; instead, the search algorithm should be applied only once.

In each of the 450 cases for each of the tests, the mission was a success. The maximum final error encountered in either test was less than 1.6 degrees, which would then be taken care of by the satellite control mechanism and so is a clear mission success. The test when the gimbal rate limit was 60 degrees per second, shown in figure 6.8, selected higher-determinant trajectories which yielded better overall tracking error. Only 1 of 450 trajectories had minimum determinant less than 0.5, and the average final eigenangle error was 0.27 degrees with a maximum final error of 0.99 degrees. In comparison,the real-time tests with gimbal rate limit of only 30 degrees per second, shown in figure 6.9, selected lower-determinant trajectories which resulted in higher (though still successful) final tracking errors. In this trial, 8 of 450 trajectories encountered a determinant less than 0.5, and the mean final eigenangle error was 0.47 degrees with a maximum final eigenangle error of 1.5 degrees. The results of these two simulations together show the promise and potential of my discrete search-based method combined with the stage division search method for real-time application. The success of the real-time trials indicates that this algorithm could be used for autonomous trajectory selection onboard a satellite.

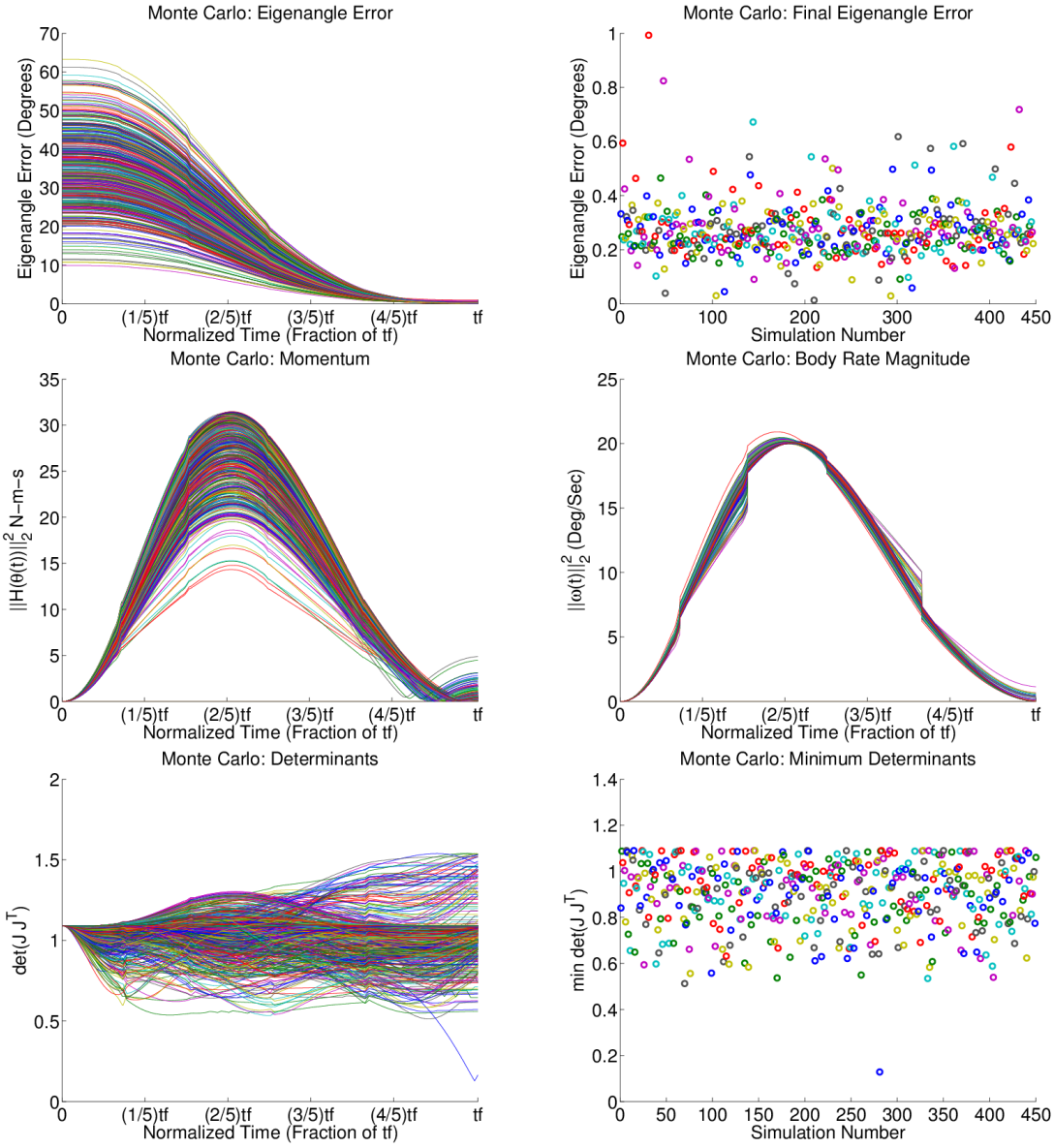


Figure 6.8: Results of 450 Monte Carlo "real-time" simulations with randomized initial and final attitudes, using stage division with 5 stages, rotation rate 10 deg/sec and $h_{CMG} = 15$ N-m-s, and gimbal rate limit 60 degrees per second. These trials all result in clear mission success.

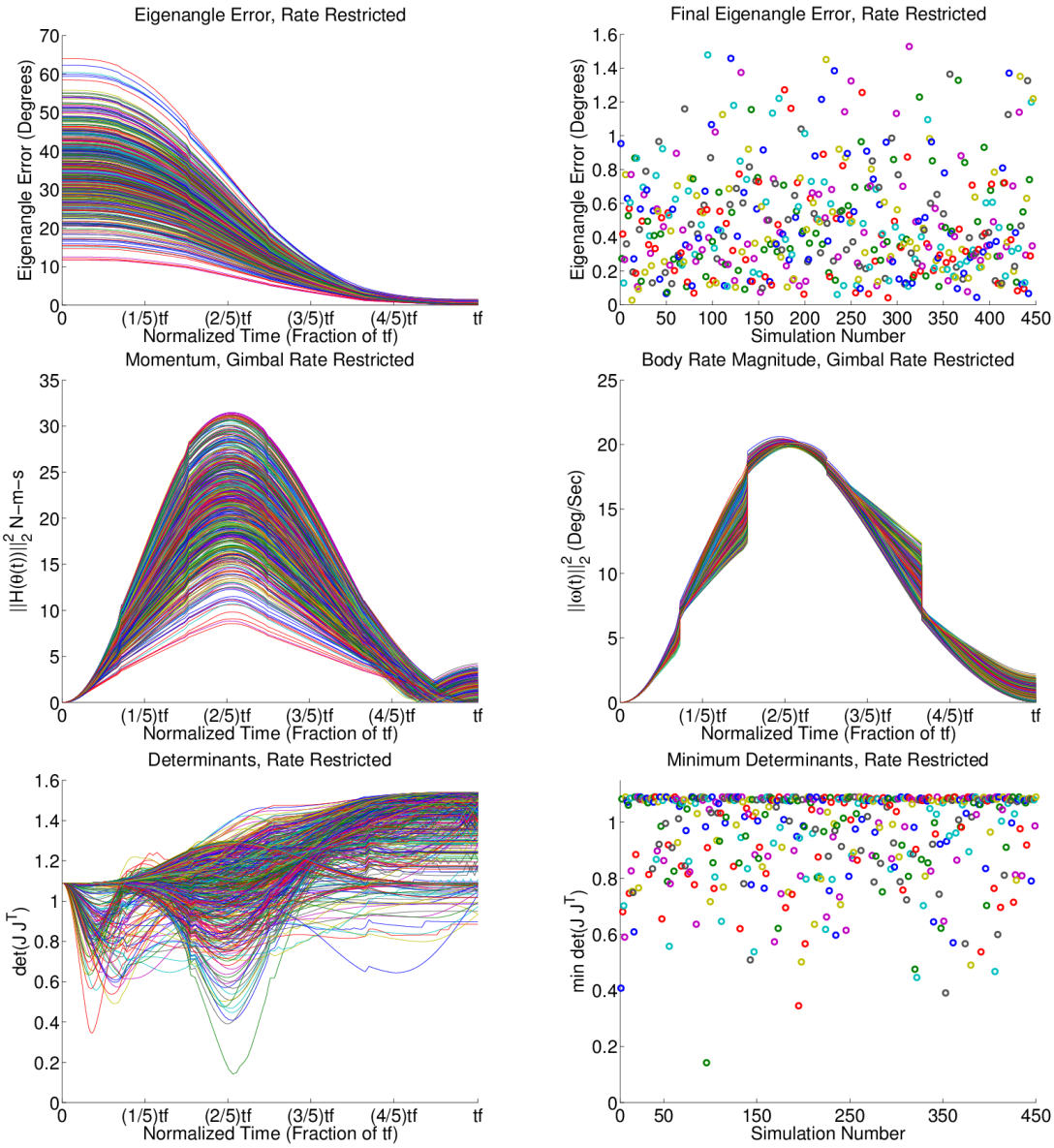


Figure 6.9: Results of 450 Monte Carlo "real-time" simulations with randomized initial and final attitudes, using stage division with 5 stages, rotation rate 10 deg/sec and $h_{CMG} = 15$ N-m-s, and gimbal rate limit 30 degrees per second. Despite some low determinants, these trials achieve high tracking accuracy.

6.4 Roll Maneuver Test Case

The most common test case for a CMG guidance algorithm in the literature is a 30 degree roll maneuver. The 30 degree roll leads the satellite directly into a singularity which is difficult to avoid, depending upon the capabilities of the satellite. For example, increasing the momentum of the individual CMGs increases the ability of the satellite to easily avoid problematic situations. In addition, singularities are easier to avoid in slow rotation compared to fast rotation. Here I compare the effects of changing the maximum gimbal rate limit of the CMGs. Since the biggest problem with the satellite having low gain is that the commanded gimbal rates can exceed the gimbal rate limit, it is interesting to observe what effect a changing gimbal rate limit has on the trajectories developed by the algorithm. With a high gimbal rate limit of 60 degrees per second as shown in the right column of figure 6.10, occurrences of low CMG gain are less problematic because the high gimbal rates required to move in regions of low CMG are still within the rate limit. When the gimbal rate limit is as low as 30 degrees per second as in the left column of figure 6.10, occurrences of low CMG gain cause tracking errors in the desired maneuver. In addition, low rate limits mean that the singularity avoidance strategy selected by the algorithm is less effective in keeping the value of CMG gain high.

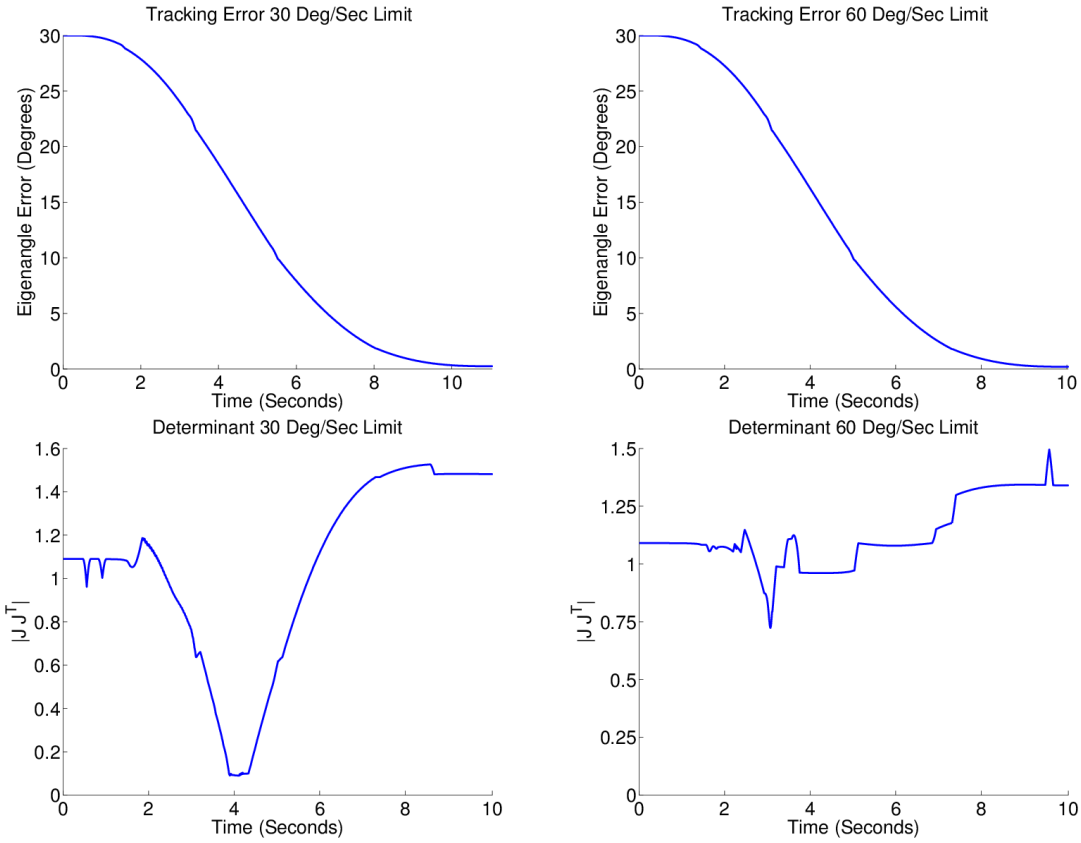


Figure 6.10: Left Column: Results for a 30 deg roll maneuver with Gimbal Rate Limit of 30 deg/sec. Right Column: The same maneuver with Gimbal Rate Limit of 60 deg/sec. Top: Tracking error for each maneuver. The 30 deg/sec trajectory has final error 0.25 degrees. The 60 deg/sec trajectory has final error 0.20 degrees. Bottom: The determinant is much worse in the 30 deg/sec trajectory than the 60 deg/sec trajectory because of stricter limitations.

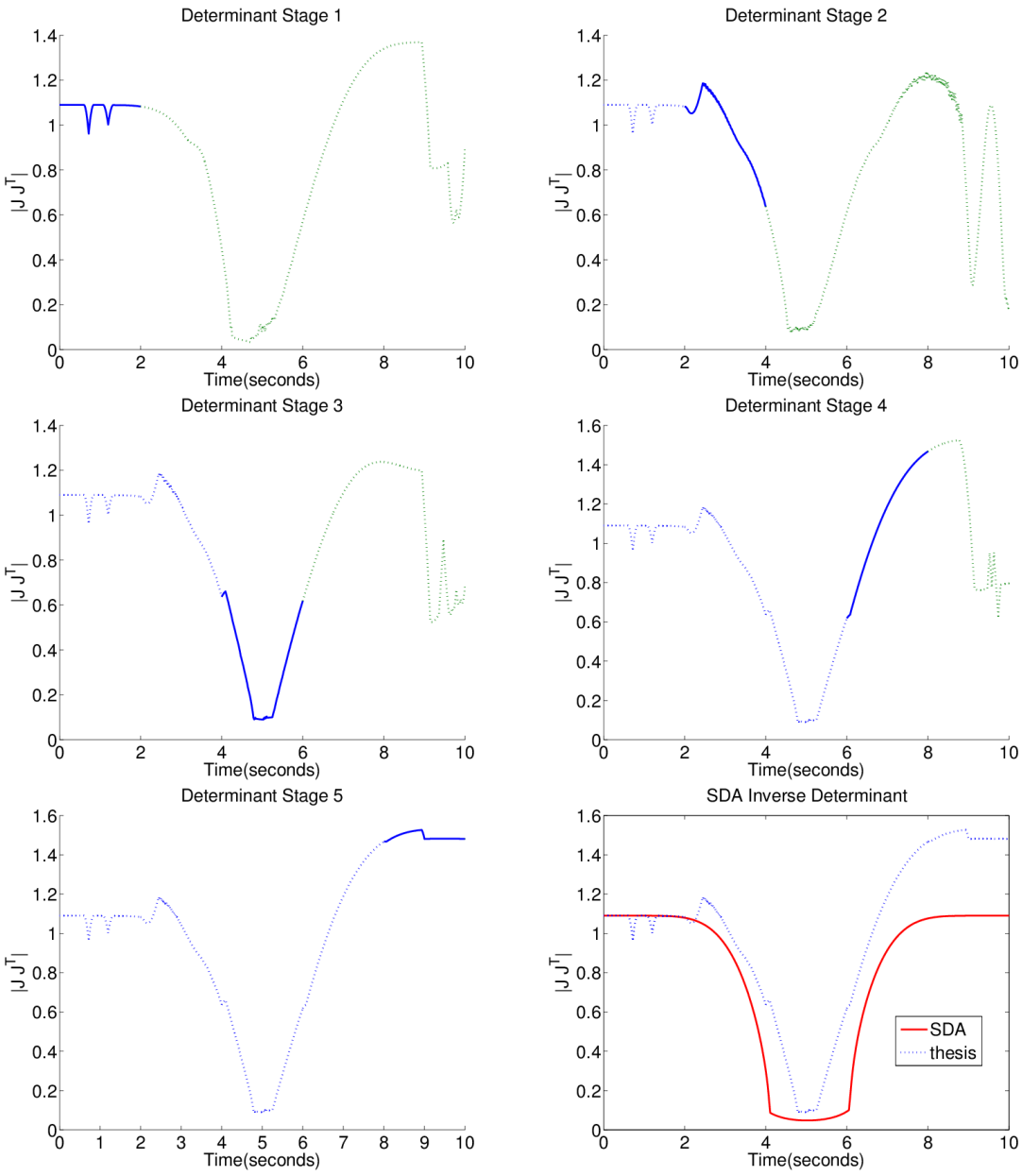


Figure 6.11: Using 5 stages, the algorithm repeats the optimization process 5 times with nodes distributed to prioritize the upcoming stage while still considering look-ahead information from the remaining stages. With a gimbal rate limit of 30 deg/sec, the algorithm struggles to avoid singularity for this 30 degree roll maneuver.

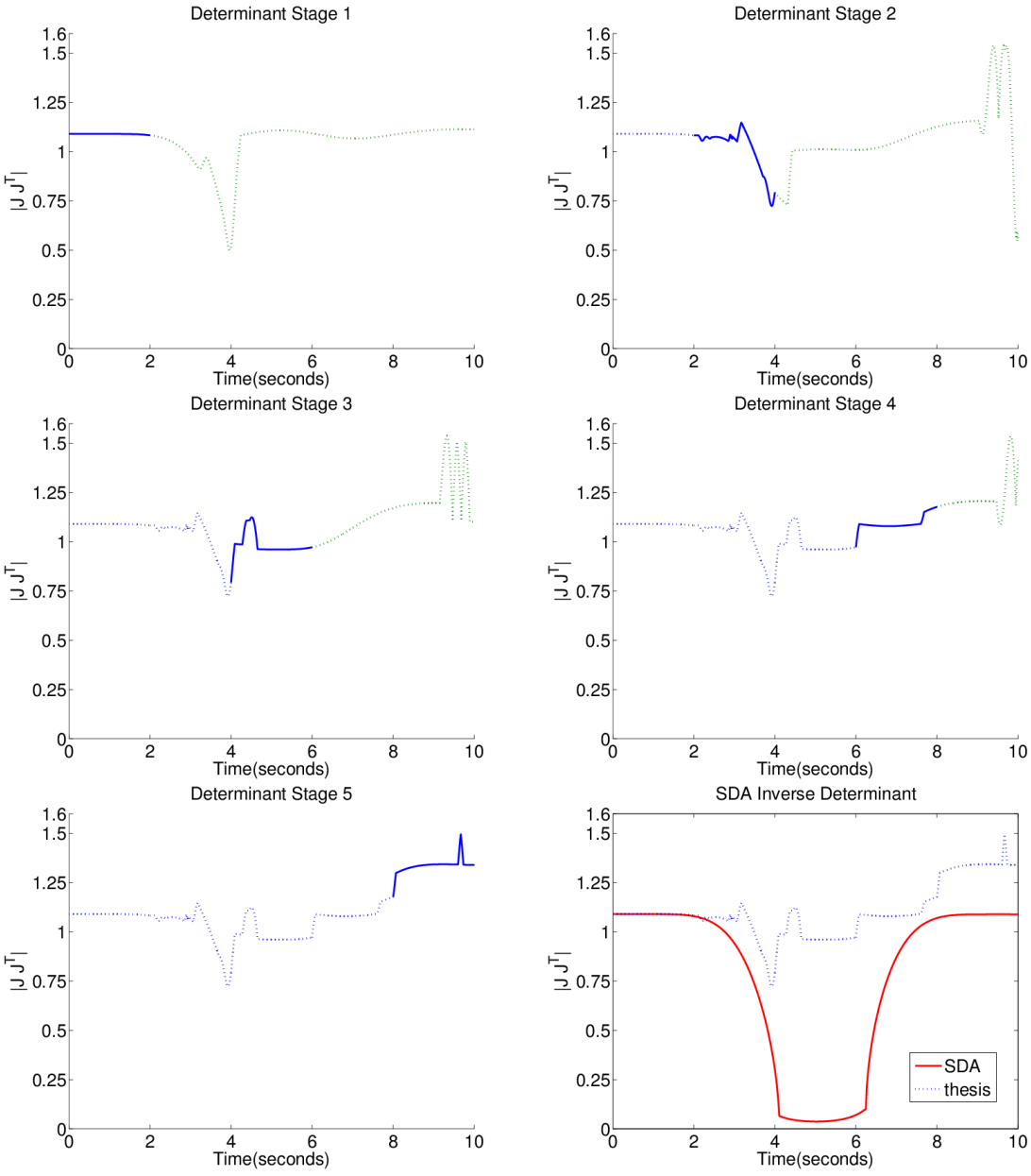


Figure 6.12: Optimization occurs 5 times, prioritizing the upcoming stage each time. With a gimbal rate limit of 60 deg/sec, the algorithm successfully manages to maintain the CMG array away from singularity for this 30 degree roll maneuver.

Chapter 7

Conclusions and Recommendations

In this thesis I successfully developed a search-based discrete optimization method, capable of providing CMG guidance for satellites to avoid singular states. Furthermore, I demonstrated through a battery of Monte Carlo simulations that the method successfully executes rotation maneuvers accurately when the capabilities of the satellite system are not exceeded, and that the method is capable of performing well under simulated real-time conditions, when computational time is limited by the actual flight time of a stage of the trajectory.

My search-based method is a combination of previously developed global solvers and recent local solvers. As a hybrid of the two, it takes advantage of the global information to provide look-ahead capability, and the singular-direction avoidance capability of the local singularity escape method. By discretizing the optimal control problem used by a global two-point boundary value solver, my method limits the

number of decisions which the satellite needs to make to choose a trajectory in order to allow satellites with limited computational capability to search over a discrete tree of potential trajectories to find near-optimal solutions to stay away from singular configurations.

By developing a real-time framework for the search algorithm, I provide a method by which a satellite with limited capability can successfully improve its trajectory to maintain its distance from singularity using its internal Control Moment Gyroscopes. The stage division method of [12] has the algorithm report a selected trajectory which is optimized for the current stage the satellite is flying, while re-optimizing to continue to improve in real time.

The success of the real-time framework is established using Monte Carlo simulations. By comparing cases with similar conditions for maneuver rates of 5, 10, and 15 degrees per second, I show that the algorithm successfully utilizes global information to improve the trajectories for the 5 and 10 degree per second cases. At 15 degrees per second, which is more demanding than the expected performance of the algorithm, the method still successfully maintained minimum CMG gain above 0.5 for 80% of tested trajectories. Additionally, I showed that the capabilities of the satellite itself have a big effect on the performance of the algorithm by comparing results when the maximum gimbal rate is 30 degrees per second to when it is the more usual 60 degrees per second. Because the most problematic effect of the satellite being in a near-singular state is that the gimbals are commanded to move at rates which exceed

their capability, the gimbal rate limit has a noticeable effect on the tracking accuracy of the trajectories provided by the algorithm. As expected, the trajectories found using a gimbal rate limit of 60 degrees per second were more accurate and had higher minimum determinants than those found using a gimbal rate limit of 30 degrees per second.

Except in the simulations which deliberately attempted to exceed the expected performance of the algorithm, my search based method successfully accomplished the satellite's rotation missions while maintaining the distance of the CMG array configuration from singularity.

The primary goal of this thesis was to develop a feasible real-time look ahead solver, and some limited comparison showed the expected improvement of this search-based method over its local parent method. Future researchers should test this method against existing methods.

Although it should be fully expected that this discrete search method will yield suboptimal results compared to a global solver using a two point boundary value solution method, results from individual trajectory simulations show that my method takes advantage of its look-ahead capability, and performs better than the SDA method which is only equipped with local information. In the future, the method could be improved by adjusting the initial exploration of the search-based algorithm.

In the method presented in this thesis, the initial exploration includes the SDA pseudoinverse solution with no null motion; the SDA pseudoinverse solutions with

constant positive and constant negative null motion; and a greedy solution where the best child node is taken at each iteration, starting from the root node. To improve the algorithm, additional initial exploration could be implemented which includes other local methods. Then, the algorithm would always be an improvement over existing methods, since it would select the best among the initial trajectories and seek to improve upon it.

Many solution methods for inverse-kinematic problems have been shared between robotics and satellite fields (see for example [19], [1]). This thesis is well-suited for application to robot manipulator control. Additionally, problems which compare to the Canadian Traveler Problem can be solved using the search-based approach developed in this thesis. This includes navigation and search applications, where a robot might have to navigate a maze by choosing to move right, left, or forward at each location.

As the computational capability of satellites continues to improve, the success of the discrete search-based solution method proposed in this thesis will continue to improve. With higher computational capability, it would be possible to consider more choices of null motion at each decision node besides the simple three choices of positive, negative, or zero. Additionally, better computational resources would allow a higher density of decision nodes in the discrete optimization, which would improve the accuracy of the discrete model to more closely match the capability of a continuous system.

Bibliography

[1] N. S. Bedrossian. Steering law design for redundant single Gimbal control moment gyro systems. *NASA STI/Recon Technical Report N*, 87:28882, August 1987.

[2] J. T. Betts. A survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March–April 1998.

[3] J. Catchpole. *The International Space Station: Building for the future*. Springer, August 2008.

[4] R. Diestel. *Graph Theory*. Springer, fourth edition, 2010.

[5] E. Edelson. Saving skylab: The untold story. *Popular Science*, 214(1):64–67,152–153, January 1979.

[6] G. D. Eppen and F. J. Gould. *Qualitative Concepts for Management: Decision Making Without Algorithms*, pages 20, pages 656–670. Prentice-Hall, Inc., second edition, 1979.

[7] F. Fahroo and I. M. Ross. User's manual for DIDO 2002: A MATLAB application package for dynamic optimization. Technical Report NPS-AA-02-002, Naval Postgraduate School, Monterey, CA, June 2002.

[8] K. A. Ford and C. D. Hall. Singular direction avoidance steering for control-moment gyros. *Journal of Guidance, Control, and Dynamics*, 23(4), July-August 2000.

[9] Ch. J. Heiberg. Momentum position control. Patent EP1188098 B1, Honeywell Inc., April 2004.

[10] P. C. Hughes. *Spacecraft Attitude Dynamics*. John Wiley and Sons, New York, 1986.

[11] J. E. Hurtado. *Kinematic and Kinetic Principles*. Lulu Marketplace, second edition, 2008.

[12] H. Ikaida, T. Tsuchiya, H. Ishii, H. Gomi, and Y. Okuno. Numerical simulation of real-time trajectory optimization for helicopter noise abatement. *Journal of Mechanical Systems for Transportation and Logistics*, 3(2):415–430, 2010.

[13] Honeywell International Inc. M50 control moment gyroscope. Catalog, Phoenix, AZ, January 2006.

[14] D. Karger and E. Nikolova. Exact algorithms for the Canadian traveller problem on paths and trees. Technical Report MIT-CSAIL-TR-2008-004, Massachusetts Institute of Technology, Cambridge, January 2008.

[15] S. Lang. *Undergraduate Analaysis*. Springer, second edition, 1997.

[16] F. A. Leve and N. G. Fitz-Coy. Hybrid steering logic for single-gimbal control moment gyroscopes. *Journal of Guidance, Control, and Dynamics*, 33(4), July-August 2010.

[17] G. Margulies and J. N. Aubrun. Geometric theory of single-gimbal control moment gyro systems. *The Journal of the Astronautical Sciences*, 26(2):159–191, April–June 1978.

[18] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789 –814, 2000.

[19] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171, September 1986.

[20] NASA. TRACE Transition Region and Coronal Explorer. `http://sunland.gsfc.nasa.gov/smex/trace/mission/trace.htm`, April 2012.

[21] Ch. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, July 1991.

[22] J. A. Paradiso. A search-based approach to steering single gimballed CMGs. Technical Report CSDL-R-2261, The Charles Stark Draper Laboratory, Inc., Cambridge, MA, August 1991.

[23] J. A. Paradiso. Global steering of single gimballed control moment gyroscopes using a directed search. *Journal of Guidance, Control, and Dynamics*, 15(5):1236–1244, 1992.

[24] J. Reeger. A comparison of transcription techniques for the optimal control of the international space station. Technical Report CSDL-T-1634, The Charles Stark Draper Laboratory, Inc., Cambridge, MA, April 2009.

[25] M. Sniedovich. *Dynamic programming: foundations and principles*. CRC Press, second edition, 2011.

[26] Ye. I. Somov, V. N. Platonov, and A. V. Sorokin. Steering the control moment gyroscope clusters onboard high-agile spacecraft. *IFAC Automatic Control in Aerospace*, pages 137–142, 2004.

[27] B. Wie. New singularity escape/avoidance steering logic for control moment gyro systems. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 11(14), August 2003. AIAA 2003-5659.

[28] D. Zimbelman and J. G. Watzin. On-orbit performance of the transition region and coronal explorer (TRACE) attitude control system. In *Guidance and Control 1999*, volume 101, pages 437–455, Breckenridge, CO, February 1999. American Astronautical Society, Advances In The Astonautical Sciences.

Appendix A

Attitude Maneuver Profile

In this section I explain how the attitude maneuver profile is generated. The mission of the satellite is to follow a particular rotation profile from an initial attitude $\mathbf{e}_0$ to a final attitude, $\mathbf{e}_f$. Although any rotation profile can be chosen, this thesis employs smooth sinusoidal profiles

$$\mathbf{e}(t)_{desired} = \left(-\frac{t_f}{2\pi}\sin\left(\frac{2\pi}{t_f}t\right)+t\right)\frac{\Delta\mathbf{e}}{t_f}, \text{ and} \tag{A.1}$$

$$\dot{\mathbf{e}}(t)_{desired} = \left(-\cos\left(\frac{2\pi}{t_f}t\right)+1\right)\frac{\Delta\mathbf{e}}{t_f}, \tag{A.2}$$

for $t \in [0\ t_f]$, where $\mathbf{e}(t)_{desired}$ is measured in radians and $\dot{\mathbf{e}}(t)_{desired}$ is measured in radians per second. Here $\Delta\mathbf{e}$ is the desired change in attitude. However, $\Delta\mathbf{e} \neq \mathbf{e}_f - \mathbf{e}_0$, since Euler angles are products of rotation matrices and are not vectors. Next I explain the process to find $\Delta\mathbf{e}$.

Let a set of Euler angles be written as $\mathbf{e} = [e_1\ e_2\ e_3]^T$. Then the corresponding Euler parameters (sometimes called quaternions in the literature) which are charac-

terized as a rotation of angle ϕ about a rotation axis defined by unit vector $\hat{\mathbf{a}}$ are given by [10]

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\phi}{2}) \\ \hat{a_1}\sin(\frac{\phi}{2}) \\ \hat{a_2}\sin(\frac{\phi}{2}) \\ \hat{a_3}\sin(\frac{\phi}{2}) \end{bmatrix} = \begin{bmatrix} \cos e_1 \cos e_2 \cos e_3 + \sin e_1 \sin e_2 \sin e_3 \\ \sin e_1 \cos e_2 \cos e_3 - \cos e_1 \sin e_2 \sin e_3 \\ \cos e_1 \sin e_2 \cos e_3 + \sin e_1 \cos e_2 \sin e_3 \\ \cos e_1 \cos e_2 \sin e_3 - \sin e_1 \sin e_2 \cos e_3 \end{bmatrix}. \tag{A.3}$$

Using equation (A.3) compute q_0 and q_f using initial attitude e_0 and final attitude e_f, respectively. Although the Euler angles are more useful for visualizing attitude, Euler parameters given by (A.3) are more useful for providing a measure of the distance between two attitudes. Let $\eta_0,\ \eta_f \in \mathbb{R}$ and let $\boldsymbol{\epsilon}_0,\ \boldsymbol{\epsilon}_f \in \mathbb{R}^3$ such that $q_0 = [\eta_0\ \boldsymbol{\epsilon}_0^T]^T$ and $q_f = [\eta_f\ \boldsymbol{\epsilon}_f^T]^T$. Then according to [10] the result of angular displacement q_0 followed by $[\eta_f\ -\boldsymbol{\epsilon}_f^T]^T$ is the eigenangle error between q_0 and q_f and is given by

$$\Delta q = \begin{bmatrix} \eta_0\eta_f - \boldsymbol{\epsilon}_0^T(-\boldsymbol{\epsilon}_f) \\ \eta_f\boldsymbol{\epsilon}_0 + \eta_0(-\boldsymbol{\epsilon}_f) + \begin{bmatrix} \epsilon_{02}(-\epsilon_{f3}) - \epsilon_{03}(-\epsilon_{f2}) \\ \epsilon_{03}(-\epsilon_{f1}) - \epsilon_{01}(-\epsilon_{f3}) \\ \epsilon_{01}(-\epsilon_{f2}) - \epsilon_{02}(-\epsilon_{f1}) \end{bmatrix} \end{bmatrix}. \tag{A.4}$$

Then using equation (A.4) it is possible to compute

$$\Delta\mathbf{e} = \begin{bmatrix} \arctan\frac{2\Delta q_0\Delta q_1 + \Delta q_2\Delta q_3}{1-2(\Delta q_1^2+\Delta q_2^2)} \\ \arcsin(2(\Delta q_0\Delta q_2 - \Delta q_3\Delta q_1)) \\ \arctan\frac{2\Delta q_0\Delta q_3 + \Delta q_1\Delta q_2}{1-2(\Delta q_2^2+\Delta q_3^2)} \end{bmatrix}. \tag{A.5}$$

This result of equation (A.5) is substituted into equations (A.1) and (A.2) to generate the complete desired body rate and acceleration profiles on $t \in [0\ t_f]$.

Next, it is necessary to find body acceleration and rate profiles $\dot{\omega}(t)_{desired}$ and $\omega(t)_{desired}$ for use in the satellite system dynamics. Using $\dot{\mathbf{e}}$ we can compute a desired body rate profile using the equation [10, p. 27]

$$\omega(t)_{desired} = S(\mathbf{e}(t)_{desired})\ \dot{\mathbf{e}}(t)_{desired}. \tag{A.6}$$

The matrix $S(\mathbf{e}(t))$ is a transformation matrix and is given by

$$S(\mathbf{e}) = \begin{bmatrix} 1 & 0 & -\sin e_2 \\ 0 & \cos e_1 & \sin e_1 \cos e_2 \\ 0 & -\sin e_1 & \cos e_1 \cos e_2 \end{bmatrix}. \tag{A.7}$$

Then using $\omega(t)_{desired}$ it is easy to generate $\dot{\omega}(t)_{desired}$. Then these profiles are used with equation (3.12) to find the desired momentum and torque profiles required by the inverse system dynamics of section 3.4.

Appendix B

Singularities

In this Appendix, I provide a derivation originally from [17] to show why null motion cannot be used to escape from some types of singularities. The purpose is to explain why torque error is inevitable as a result of approaching certain types of singularities. At the end of the chapter are two figures which show singular configurations.

While in a singular configuration, it is impossible to produce torque in the singular direction. When the rank of $J(\theta(t)) = 2$, the range of the Jacobian is a plane. For $\dot{H}(t)_{desired} \notin R\Big(J(\theta(t))\Big)$, there is no $\dot{\theta}(t)$ which satisfies (3.13). Since (3.13) is used extensively by existing local methods and the discrete search method of this thesis, there are two "escape mechanisms" to allow the guidance methods to trade the disadvantages of tracking error for the hazards of getting trapped in a singular state: torque error and null motion.

Torque error means intentionally commanding a value of $\dot{\theta}(t)$ such that

$$\dot{H}(t)_{desired} \neq J(\theta(t))\dot{\theta}(t). \tag{B.1}$$

Torque error is applied as the system approaches singularity to prevent the system from actually entering a singular configuration, which causes even greater error as the satellite exhibits unpredictable behavior. Torque error prevents the particular solution from commanding unachievable gimbal rates and slows the system response to allow more torque error to be applied [1]. The torque error can be corrected by the control mechanism of the satellite after skirting the singularity.

The more desirable singularity avoidance mechanism is null motion. If we let $\dot{\theta}_{null}(t) \in N\Big(J(\theta(t))\Big)$ where $N\Big(J(\theta(t)\Big)$ is the null space of the Jacobian, then commanding the satellite gimbals to follow $\dot{\theta}_{null}(t)$ does not produce any torque on the satellite. This is known as null motion, and it is the more desirable method of avoiding or escaping singular configurations; when null motion is applied,

$$\dot{H}(\theta(t)) = J(\theta(t))\dot{\theta}_{null} = 0. \tag{B.2}$$

Since the Jacobian of the system is 3×4 the null space always has dimension of at least one. However, there are two types of singularities, near one of which null motion cannot be used as a singularity escape method, as shown in the derivation in the next section.

B.1 Classification of Singularities

By performing a Taylor series expansion about a singular configuration $\boldsymbol{\theta}^s$, it is possible to investigate the types and behavior of various types of singularities. This is intended to be more accessible than derivation in [16]. Assume that a small movement $\Delta\boldsymbol{\theta}$ is made to a new configuration $\boldsymbol{\theta}$. The Taylor series expansions about each scalar element of $\mathbf{H}(\boldsymbol{\theta}^s) = [\ h_1 \quad h_2 \quad h_3\ ]^T$ are

$$h_1(\boldsymbol{\theta}) - h_1(\boldsymbol{\theta}^s) = \nabla h_1(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^T\nabla^2 h_1(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + O(\|\Delta\boldsymbol{\theta}\|^3) \quad \text{(B.3)}$$

$$h_2(\boldsymbol{\theta}) - h_2(\boldsymbol{\theta}^s) = \nabla h_2(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^T\nabla^2 h_2(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + O(\|\Delta\boldsymbol{\theta}\|^3) \quad \text{(B.4)}$$

$$h_3(\boldsymbol{\theta}) - h_3(\boldsymbol{\theta}^s) = \nabla h_3(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^T\nabla^2 h_3(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + O(\|\Delta\boldsymbol{\theta}\|^3). \quad \text{(B.5)}$$

By combining these Taylor series expansions into vector form and noting from equation (3.7) that the off-diagonal terms of $\nabla^2 h_1(\boldsymbol{\theta}^s)$, $\nabla^2 h_2(\boldsymbol{\theta}^s)$, and $\nabla^2 h_3(\boldsymbol{\theta}^s)$ are zero, the expansion becomes

$$\begin{aligned}\mathbf{H}(\boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\theta}^s) &= \nabla\mathbf{H}(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + \frac{1}{2}\begin{bmatrix} \Delta\boldsymbol{\theta}^T\nabla^2 h_1(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} \\ \Delta\boldsymbol{\theta}^T\nabla^2 h_2(\boldsymbol{\theta}^s)\Delta\theta \\ \Delta\boldsymbol{\theta}^T\nabla^2 h_1(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} \end{bmatrix} + O(\|\Delta\boldsymbol{\theta}\|^3) \\ &\cong \nabla\mathbf{H}(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} + \frac{1}{2}(\Delta\theta_1^2\frac{\delta^2\mathbf{H}}{\delta\theta_1^2} + \Delta\theta_2^2\frac{\delta^2\mathbf{H}}{\delta\theta_2^2} + \Delta\theta_3^2\frac{\delta^2\mathbf{H}}{\delta\theta_3^2} + \delta\theta_4^2\frac{\delta^2\mathbf{H}}{\delta\theta_4^2}).\end{aligned}$$

From the definition of the momentum $\mathbf{H}$ in equation (3.7) and using the notation $\mathbf{H} = \mathbf{H}_1 + \mathbf{H}_2 + \mathbf{H}_3 + \mathbf{H}_4$, to define the total momentum $\mathbf{H}$ as the sum of the individual momenta of each CMG, it is easy to see that $\frac{\delta^2\mathbf{H}}{\delta\theta_1^2} = -\mathbf{H}_1$, $\frac{\delta^2\mathbf{H}}{\delta\theta_2^2} = -\mathbf{H}_2$, and so on. Recalling the definition of the Jacobian of the momentum, the previous

equation becomes

$$H(\boldsymbol{\theta}) - H(\boldsymbol{\theta}^s) \cong J(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} - \frac{1}{2}(\Delta\theta_1^2\mathbf{H}_1 + \Delta\theta_2^2\mathbf{H}_2 + \Delta\theta_3^2\mathbf{H}_3 + \Delta\theta_4^2\mathbf{H}_4). \tag{B.6}$$

Next, project this equation into the singular direction. Since $\boldsymbol{\theta}^s$ is assumed to be a singular point, the null space of the Jacobian at that point has rank two and so there is exactly one singular direction. Using a standard Singular Value Decomposition $USV^T = J(\boldsymbol{\theta}^s)$, the smallest singular value occurs in the third column of S; and its corresponding eigenvector $\mathbf{u}$ is the singular direction. Then

$$[H(\boldsymbol{\theta}) - H(\boldsymbol{\theta}^s)] \cdot \mathbf{u} \cong J(\boldsymbol{\theta}^s)\Delta\boldsymbol{\theta} \cdot \mathbf{u} - \frac{1}{2}(\Delta\theta_1^2\mathbf{H}_1 \cdot \mathbf{u} + \Delta\theta_2^2\mathbf{H}_2 \cdot \mathbf{u} + \Delta\theta_3^2\mathbf{H}_3 \cdot \mathbf{u} + \Delta\theta_4^2\mathbf{H}_4 \cdot \mathbf{u}). \tag{B.7}$$

Assume that $\Delta\boldsymbol{\theta}$ is null motion and moves the system from a singular to a non-singular state. From the definition of null motion, the first term on the right hand side of equation (B.7) is zero. Since the movement $\Delta\boldsymbol{\theta}$ was null motion, no torque was produced; so $\mathbf{H}(\boldsymbol{\theta}) = \mathbf{H}(\boldsymbol{\theta}^s)$. Define $P = diag([\mathbf{H}_1 \cdot \mathbf{u}\ \mathbf{H}_2 \cdot \mathbf{u}\ \mathbf{H}_3 \cdot \mathbf{u}\ \mathbf{H}_4 \cdot \mathbf{u}])$ and then equation (B.7) becomes

$$0 \cong \Delta\boldsymbol{\theta}^T P \Delta\boldsymbol{\theta}. \tag{B.8}$$

Since $\boldsymbol{\theta}^s$ is a singular configuration the null space has rank two. Define $\boldsymbol{\gamma} \in \mathbb{R}^2$ and let $N \in \mathbb{R}^{4\times 2}$ be a basis for the null space of $J(\boldsymbol{\theta}^s)$. Then $\Delta\boldsymbol{\theta} = N\boldsymbol{\gamma}$, and

$$0 = \boldsymbol{\gamma}^T N^T P N \boldsymbol{\gamma} = \boldsymbol{\gamma}^T Q \boldsymbol{\gamma}, \tag{B.9}$$

where $Q = N^T P N$ has rank 2 when the system is singular and rank 1 when the system is nonsingular. At this point in the derivation, we have assumed that the

system is at a singular state $\boldsymbol{\theta}^s$ and that we were able to move a small distance $\Delta\boldsymbol{\theta}$ to a nonsingular state $\boldsymbol{\theta}$. Now we will examine Q to see under what conditions those assumptions are possible. So we will examine two cases: the matrix Q is definite; it is semidefinite or indefinite.

If the matrix is definite, all its eigenvalues are positive or they are all negative. So it is impossible to find a non-zero scaling vector $\boldsymbol{\gamma}$ such that equation (B.9) is satisfied. So the assumption that we escaped the singular state $\boldsymbol{\theta}^s$ using null motion was false. Therefore we conclude that when the matrix Q is definite, it is impossible to use motion in the null space to achieve a nonsingular configuration. This type of singularity is given the name elliptic. These types of singularities are the most troublesome for high-precision satellite maneuvers, because escaping them requires inducing an unwanted torque error on the satellite. Using a top-down view of the pyramid mount shown in figure 3.2, an example of an elliptic singularity is shown in figure B.1. Note that all the momentum vectors are in line with the singular direction. It is impossible to use null motion to escape the singular state shown.

If Q is semidefinite, at least one of its eigenvalues is nonzero, so there are infinitely many nonzero vectors $\boldsymbol{\gamma}$ which satisfy equation (B.9). If it is indefinite, it has one positive and one negative eigenvalue; so there exist nonzero vectors which can satisfy (B.9). Therefore it is possible to find a scaling vector $\boldsymbol{\gamma}$ such that equation (B.9) is satisfied. So in the case where Q is semidefinite or indefinite, it is possible to escape from a singular to a non-singular state using null motion. These types of singularities

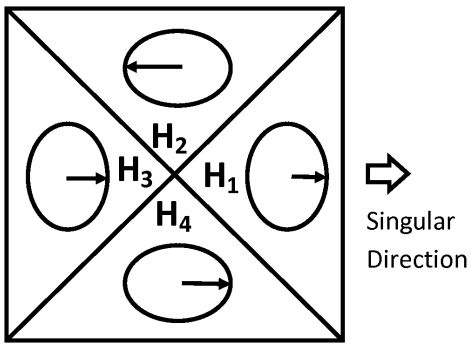


Figure B.1: From [1]. Elliptic singularity with with $\theta = [-90^o\ 0^o\ 90^o\ 0^o]$. It is impossible to produce torque in the singular direction, and no null motion is possible to escape the singular state.

are known as hyperbolic singularities because they can be escaped without causing a torque disturbance. An example of a hyperbolic singularity is shown in figure B.2. It is easy to visualize the null motion which could be used to change the singular state shown in figure B.2 to a nonsingular state: by rotating H_1 and H_3 clockwise and counter-clockwise, respectively, the momentum of the system is unchanged and it becomes possible to create torque in the singular direction.

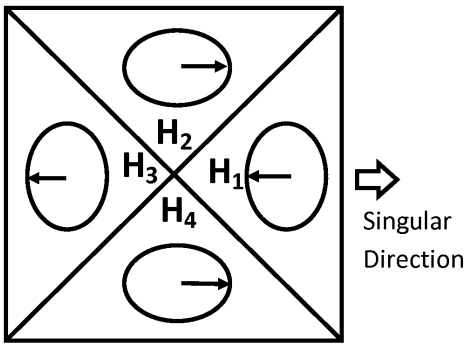


Figure B.2: Adapted from [1]. Hyperbolic singularity with $\theta = [90^o\ 180^o\ -90^o\ 0^o]$. It is impossible to produce torque in the singular direction, but null motion can be used to escape the singular state by counter-rotating gimbals 1 and 3.